بسم الله الرحمن الرحیم

**مستند سازی پروژه MLOps**

**شرکت آدین**

****

**اسفند ۱۴۰۱**

**نسخه ۰.۵**

فهرست

[۱. مشخصات کلی............................................................................................................................4](#_Toc127901742)

[نکات مثبت MLOP 5](#_Toc127901743)

[معایب MLOps 5](#_Toc127901744)

[Kubeflow چیست؟ 5](#_Toc127901745)

[معماری سرویس Kubeflow 6](#_Toc127901746)

[ویژگی های سرویس Kubeflow 6](#_Toc127901747)

[مقایسه Kubeflow و MLflow 7](#_Toc127901748)

[نمونه های جهانی استفاده از Kubeflow 7](#_Toc127901749)

[۲. معماری و تکنولوژی‌های مرتبط](#_Toc127901750) ...........................................................................................................8

[معماری مفهومی 8](#_Toc127901751)

[مرحله آماده سازی داده‌ها در چرخه حیات ML 9](#_Toc127901752)

[معماری منطقی 9](#_Toc127901753)

[معماری فیزیکی 10](#_Toc127901754)

[بهترین ابزار منبع باز برای جمع‌آوری داده ها 10](#_Toc127901755)

[ذخیره سازی داده‌ها 11](#_Toc127901756)

[بهترین دریاچه داده برای MLOps 14](#_Toc127901757)

[3. ویژگی‌های کارکردی](#_Toc127901758)

[شکاف‌های ارتباطی تیم‌های تجاری و فنی 16](#_Toc127901759)

[ارزیابی ریسک 16](#_Toc127901760)

[بازتاب تغییر اهداف کسب و کار در مدل 16](#_Toc127901761)

[۴. ویژگی‌های غیرکارکرد16.……………………………………………………………………………………………………………………](#_Toc127901762)

[آمادگی hybrid-cloud 16](#_Toc127901763)

[امنیت 16](#_Toc127901764)

[مقیاس پذیری 16](#_Toc127901765)

[تحمل خطا 16](#_Toc127901766)

[ارائه توان عملیاتی بالا 16](#_Toc127901767)

[پاسخ سریع 16](#_Toc127901768)

[۵. فرضیات و محدودیت‌ها ...................................................................................................................16](#_Toc127901769)

[۶. تست و ارزیابی MLOPs..................................................................................................................17](#_Toc127901770)

[تست مدل 17](#_Toc127901771)

[تست برنامه 17](#_Toc127901772)

[۷. آموزش و تحویل نصب و راه اندازی MLOPs............................................................................................17](#_Toc127901773)

[آموزش نصب Kubeflow 18](#_Toc127901774)

[راه اندازی Kubeflow 18](#_Toc127901775)

[داشبورد Kubeflow 22](#_Toc127901776)

[بخش های مختلف kubeflow به همراه نحوه کار آنها 22](#_Toc127901777)

[خط لوله kubeflow 25](#_Toc127901778)

[ایجاد و بارگذاری خط لوله 29](#_Toc127901779)

[تجربه اجرای یک خط لوله (Experiments kfp) 32](#_Toc127901780)

[Endpoint 33](#_Toc127901781)

[Kserve 33](#_Toc127901782)

[۸. منابع موردنیاز......................................................................................................................36](#_Toc127901783)

[۹. زمان‌بندی و شکست کارها........................................................................................................36](#_Toc127901784)

[۱۰. تحلیل هزینه.....................................................................................................................37](#_Toc127901785)

# مشخصات کلی

مدیریت سیستم‌های هوش مصنوعی در مقیاس بزرگ کار آسانی نیست و در این مسیر چالش‌های مهمی وجود دارد که تیم‌ها باید با آن مواجه شوند. برخی از این چالش‌ها در کاربرد یادگیری ماشین (ML) در یک محصول مقیاس‌پذیر عبارتند از:

* کیفیت و کمیت داده: اطمینان از وجود داده‌های آموزشی باکیفیت بالا برای آموزش مدل‌های ML یک چالش بزرگ است.
* عملکرد مدل: مدل‌های ML گاهی اوقات ممکن است برای تعمیم به داده‌های جدید مشکل داشته باشند و ممکن است در صورت استقرار در تنظیمات دنیای واقعی ضعیف عمل کنند.
* تفسیرپذیری مدل: مدل‌های ML می‌توانند پیچیده و غیرقابل‌درک باشند، و تفسیر پیش‌بینی‌ها و تشخیص مشکلات، آن‌ها را به چالش می‌کشد.
* منابع محاسباتی: آموزش و به‌کارگیری مدل‌های ML بزرگ و پیچیده می‌تواند به مقدار قابل‌توجهی از منابع محاسباتی نیاز داشته باشد که می‌تواند در یک محصول مقیاس‌پذیر چالش‌برانگیز باشد.
* استقرار مدل: استقرار مدل‌های ML در یک محیط تولید و ادغام آن‌ها در یک محصول می‌تواند چالش‌برانگیز باشد و نیاز به بررسی دقیق زیرساخت‌ها، حریم خصوصی داده‌ها و امنیت دارد.
* نگهداری مدل: مدل‌های ML باید با تغییر داده‌ها و نیازمندی‌های محصول، به‌روزرسانی و بازآموزی شوند که می‌تواند در یک محصول مقیاس‌پذیر یک چالش باشد.
* بایاس: مدل‌های ML گاهی اوقات می‌توانند سوگیری‌های موجود را در داده‌های آموزشی را تداوم بخشند و توجه به این مسائل هنگام توسعه یک محصول مقیاس‌پذیر ML بسیار مهم خواهد بود.

MLOps (عملیات یادگیری ماشین) به مجموعه‌ای از فرایندها، ابزارها و شیوه‌ها جهت مدیریت چرخه مدل‌های یادگیری ماشین در یک محیط تولید، اشاره دارد. همچنین این چرخه شامل همکاری بین دانشمندان داده، مهندسان و تیم‌های DevOps است به‌نحوی‌که این اطمینان حاصل شود که مدل‌ها به طور مؤثر توسعه، استقرار، نظارت و به‌روزرسانی می‌شوند. هدف MLOها افزایش سرعت، قابلیت اطمینان و مقیاس‌پذیری مدل‌های یادگیری ماشین در تولید است، درحالی‌که خطرات ناشی از ریسک عدم موفقیت را نیز کاهش می‌دهد. همچنین به‌کارگیری MLOps فرایند مدیریت را ساده‌تر کرده، کیفیت را افزایش می‌دهد و استقرار مدل‌های یادگیری عمیق و یادگیری ماشین را در محیط‌های تولید در مقیاس بزرگ را خودکار می‌کند. هدف، بهبود خودکارسازی و ارتقای کیفیت مدل‌های تولید و درعین‌حال توجه به الزامات تجاری و نظارتی است.

MLOps به‌ویژه در مورد استقرار مدل‌های ML در تولید اهمیت دارد، زیرا به سازمان‌ها کمک می‌کند تا مطمئن شوند که مدل‌هایشان در طول زمان دقیق، قابل‌اعتماد و کارآمد هستند. به‌طورکلی، MLOps با خودکار کردن بسیاری از مراحل مربوط به استقرار و مدیریت مدل‌های ML، به دانشمندان و مهندسان داده اجازه می‌دهد تا با همکاری یکدیگر به ارائه سریع‌تر و کارآمدتر مدل‌های یادگیری ماشین دست یابند. برای رسیدن به اهداف فوق، تیم‌های MLOps معمولاً از ترکیبی از ابزارها، فرایندها و شیوه‌ها زیر را استفاده می‌کنند:

* کنترل نسخه برای مدل‌های ML و مجموعه‌داده‌ها
* آزمون خودکار و ادغام مداوم
* کانتینرسازی و ارکستراسیون برای استقرار مدل
* نظارت و ثبت گزارش برای ردیابی عملکرد مدل در تولید
* حاکمیت و انطباق برای مدل‌ها

## نکات مثبت MLOP

* بهبود سرعت ایجاد و استقرار مدل در نتیجه مدیریت جامع چرخه زندگی یادگیری ماشین، به دلیل افزایش ارزش یادگیری ماشین.
* MLOها همکاری و شفافیت را در میان تیم‌های dataOps، مهندسین یادگیری ماشین، تحلیلگران تجاری/مدیران محصول، مهندسین تضمین کیفیت و مهندسین زیرساخت افزایش می‌دهند.
* MLOps زمان بازاریابی جهت الگوریتم‌های یادگیری ماشین را کاهش می‌دهد و روش‌های یکپارچه‌سازی و تحویل مداوم، استقرار این سیستم‌ها را در تولید آسان‌تر می‌کند. جهت اطمینان از ارائه پیش‌بینی‌های باکیفیت بالا، سیستم MLOps باید قادر به اندازه‌گیری دریفت مدل[[1]](#footnote-1) باشد. این امر احتمال دیدگاه‌های نادرست در فرضیه های مدل را کاهش می‌دهد.
* بهبود کیفیت مدل: با خودکارسازی آموزش، اعتبارسنجی و استقرار مدل‌ها، احتمال خطای انسانی کاهش می‌یابد که منجر به عملکرد بهتر مدل می‌شود.
* زمان سریع‌تر جهت بازاریابی: MLOps دانشمندان و مهندسان داده را قادر می‌سازد تا سریع‌تر تکرار کنند و مدل‌ها را سریع‌تر به تولید بفرستند.
* همکاری بهتر: MLOps همکاری بین دانشمندان داده، توسعه‌دهندگان و تیم‌های عملیاتی را تشویق می‌کند و تیم‌های متقابل را قادر می‌سازد تا به طور مؤثرتری با هم کار کنند.
* مقیاس‌پذیری: خودکار کردن فرایند استقرار، مقیاس‌پذیری مدل‌ها را جهت رسیدگی به مقادیر بزرگ و افزایشی داده و جمعیت بیشتر کاربران آسان‌تر می‌کند.

## معایب MLOps

* **پیچیدگی**: پیاده‌سازی MLOها می‌تواند پیچیده باشد و به مهارت‌های تخصصی و درک عمیق فرایندهای یادگیری ماشین و توسعه نرم‌افزار نیاز دارد. MLOps به ابزارها و مهارت‌های فراوانی نیاز دارد که به پیچیدگی آن می‌افزاید. جهت ساختن یک چارچوب موفق MLOps، به یک معمار باتجربه MLOps و مهندسان ML نیاز داریم که از قبل با اصول مفهوم DevOps آشنا هستند.
* **چالش‌های یکپارچه‌سازی**: ادغام جریان‌های کاری ML در فرایندهای توسعه نرم‌افزار موجود می‌تواند دشوار باشد و ممکن است نیاز به تغییرات قابل‌توجهی در جریان‌های کاری و ابزارهای موجود داشته باشد.
* **امنیت داده‌ها**: MLOها به مدیریت دقیق داده‌های حساس از جمله ذخیره‌سازی ایمن، کنترل دسترسی و نظارت بر استفاده از داده‌ها نیاز دارند.
* **محدودیت‌های منابع:** پیاده‌سازی MLOها نیازمند سرمایه‌گذاری قابل‌توجهی در فناوری، فرآیندها و پرسنل است که ممکن است جهت همه سازمان‌ها امکان‌پذیر نباشد. MLOps جهت استقرار مؤثر مدل ML در دنیای واقعی ضروری است، اما هزینه آن به دلیل پیچیدگی و منابع موردنیاز می‌تواند بسیار گران باشد. توانایی ایجاد و حفظ یک چرخه حیات MLOps به طور مؤثر به منابع کافی (افراد، زیرساخت‌ها و ابزارها) نیاز دارد. عامل اصلی تعیین‌کننده در استفاده یا عدم استفاده از MLO ها، بررسی کامل بازگشت سرمایه ابتکار یادگیری ماشین است.

## Kubeflow چیست؟

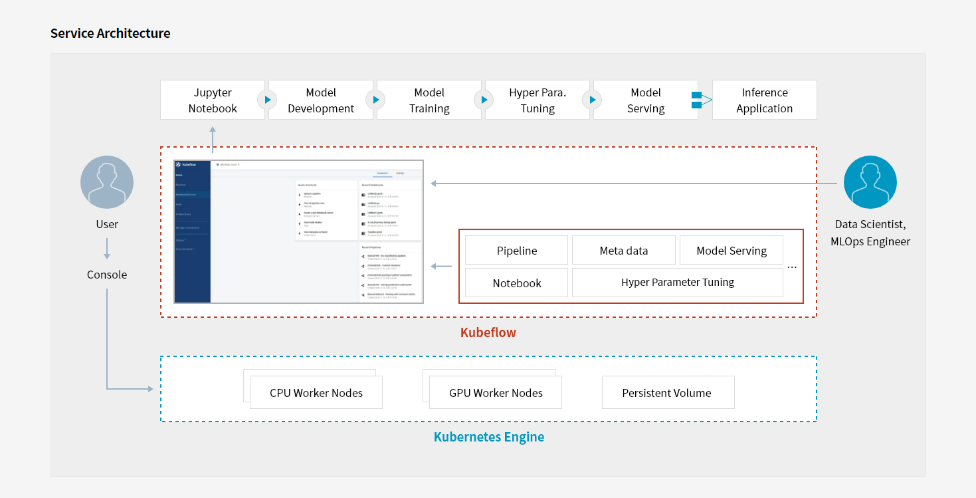
Kubeflow یک پلتفرم منبع باز برای توسعه، استقرار و مدیریت مدل‌های یادگیری ماشینی (ML) در Kubernetes (سیستم ارکستراسیون کانتینر) است به نحوی که توسعه، آزمایش و استقرار مدل‌های ML را در محیط‌های تولید برای سازمان‌ها آسان کرده و کارایی و مقیاس‌پذیری گردش‌های کاری ML خود را بهبود بخشند.

ویژگی‌های کلیدی Kubeflow عبارتند از:

* یک رابط کاربر پسند برای آموزش و استقرار مدل‌های ML با استفاده از چارچوب‌های مطرح مانند TensorFlow، PyTorch و scikit-learn و ...
* پشتیبانی از آموزش توزیع‌شده مدل با استفاده از Kubernetes، که امکان آموزش سریع‌تر و استفاده بهتر از منابع را فراهم می‌کند.
* ابزارهایی برای ثبت و تجسم پیشرفت آموزش مدل، و همچنین برای تنظیم هایپرپارامتر و استقرار مدل.
* ادغام با کتابخانه‌ها و ابزارهای معروف یادگیری ماشین، مانند TensorFlow Extended (TFX) و Seldon و ...
* توانایی اجرای در سیستم شخصی، در فضای ابری یا در محیط های ترکیبی.
* به هدف استقرار کلیه کامپوننت های چرخه یادگیری ماشین بر بستر کوبرنتیز ساخته شده است. از این رو میتوان گفت یک مجموعه ابزار کاملا منطبق و همگام با زیست بوم ابری میباشد. همچنین برای CNCF و دریافت گواهینامه معتبر زیست بوم ابری این نهاد درخواست فرستاده است.

به‌طورکلی، Kubeflow یک پلت فرم قدرتمند و منعطف برای مدیریت چرخه یادگیری ماشین است که به دانشمندان و مهندسان داده اجازه می‌دهد تا با هم کار کنند تا مدل‌های یادگیری ماشینی خود را به شیوه‌ای کارآمدتر و مؤثرتر بسازند، مستقر کنند و مدیریت کنند.

## معماری سرویس Kubeflow



## ویژگی های سرویس Kubeflow

**محیط MLOps** **مبتنی بر ابر**

Kubeflow محیط‌های توسعه مدل ML را بهینه‌سازی شده برای ابر ارائه می‌دهد و امکان پیوند مبتنی بر Kubernetes را با نرم‌افزارهای منبع باز مختلف فراهم می‌کند.

**راحتی استفاده برای داده های بزرگ**

محیط های استاندارد از طیف وسیعی از چارچوب های یادگیری ماشین از TensorFlow، PyTorch، scikit-learn و Keras پشتیبانی می کنند. خط لوله برای کل فرایندهای توسعه، یادگیری و استقرار مدل‌های یادگیری ماشینی خودکار است تا از پیکربندی/ایجاد ساده و همچنین استفاده مجدد از مدل‌ها اطمینان حاصل شود.

**ویژگی های افزودنی**

Kubeflow منبع باز گسترده‌ای از اجرای کار یادگیری توزیعی/نظارت تا مدیریت خدمات استنتاج/تحلیل و مدیریت صف شغل در هر پلتفرمی ارائه می‌دهد. کاربران همچنین می‌توانند از زمان‌بندی‌های کار (FIFO، Bin-packing و Gang-based)، نظارت بر منابع GPU، گزارش موتور Kubeflow و ویژگی‌های اضافی دیگری که معمولاً در نرم‌افزارهای منبع باز در دسترس نیستند، استفاده کنند.

- سایر ویژگی ها

* نوت بوک Jupyter (توسعه مدل، یادگیری و استنتاج)
* اتوماسیون گردش کار (بر اساس خطوط لوله یادگیری ماشین)
* مدیریت زمانبندی کار GPU و صف کار
* نظارت بر منابع GPU
* مانیتورینگ/ ثبت نام Kubeflow و اجرای/نظارت کار یادگیری توزیع شده
* ساخت و مدیریت چارچوب ML (TensorFlow، PyTorch و غیره)
* مدیریت/تجزیه و تحلیل خدمات استنتاج و مدیریت آزمایش های مدل/گره های یادگیری

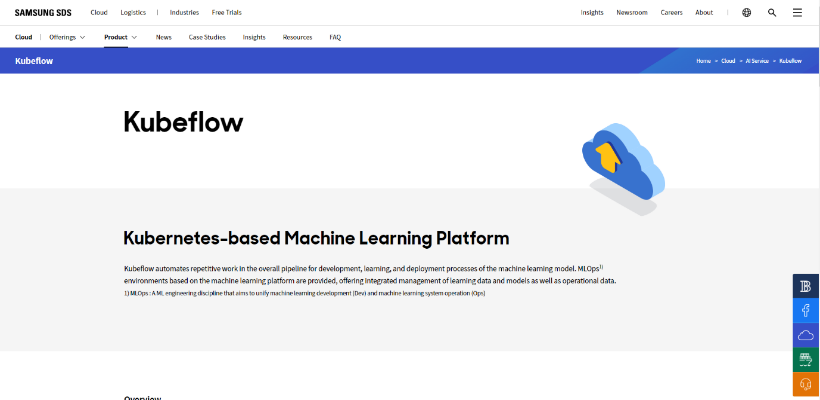
## مقایسه Kubeflow و MLflow

Kubeflow و MLflow هر دو پروژه های منبع باز برای مدیریت گردش کار یادگیری ماشین هستند. با این حال، بررسی آنها در ابعاد مختلف نشان‌گر تفاوت‌های اساسی میان این دو پروژه است:

* هدف: Kubeflow به طور خاص برای اجرا و استقرار مدل‌های یادگیری ماشین در Kubernetes، یک سیستم منبع باز محبوب برای خودکارسازی استقرار، مقیاس‌بندی و مدیریت برنامه‌های کاربردی کانتینری طراحی شده است. از سوی دیگر، MLflow یک پلت فرم کلی تر برای مدیریت کل چرخه زندگی یادگیری ماشینی است، از جمله آزمایشات ردیابی، کد بسته بندی در اجراهای تکرارپذیر، و به اشتراک گذاری و استقرار مدل ها.
* ویژگی‌ها: Kubeflow مجموعه‌ای از ابزارها و APIها را برای ساخت، استقرار و مدیریت مدل‌های یادگیری ماشین در Kubernetes ارائه می‌کند، از جمله نوت‌بوک‌های Jupyter، آموزش TensorFlow و سرویس‌دهی مدل. MLflow طیف گسترده‌تری از ویژگی‌ها را برای چرخه زندگی یادگیری ماشین ارائه می‌کند، از جمله ردیابی آزمایش، بسته‌بندی کد و اشتراک‌گذاری، و استقرار مدل در پلتفرم‌های مختلف از جمله Docker، AzureML، و Amazon SageMaker.
* یکپارچه سازی: Kubeflow به خوبی با سایر اجزای اکوسیستم Kubernetes، مانند Helm (یک مدیر بسته برای Kubernetes) ادغام می شود، و همچنین ابزارهایی را برای کاهش حجم کاری یادگیری ماشین در مجموعه ای از ماشین ها فراهم می کند. MLflow را می توان با انواع کتابخانه ها و پلتفرم های یادگیری ماشین استفاده کرد و یک REST API برای ادغام با ابزارهای دیگر ارائه می دهد.

به طور خلاصه، Kubeflow برای مدیریت بارهای کاری یادگیری ماشین در Kubernetes مناسب است، در حالی که MLflow یک پلتفرم کلی تر برای مدیریت کل چرخه زندگی یادگیری ماشین است.

## نمونه های جهانی استفاده از Kubeflow

* از جمله شرکت ها و سازمان هایی که از Kubeflow به طور جدی برای آسان سازی فرانید چرخه یادگیری ماشین استفاده می‌کنند میتوان به مجموعه ای در شرکت سامسونگ با نام Samsung SDS اشاره کرد. در واقع این مجموعه Kubeflow را به عنوان پلتفرم ابری منطبق باKubernetes انتخاب کرده و به مشتریان خود بر بستر Samsung Cloud Platform پیاده‌سازی و خدمات ارائه میدهد.[[2]](#footnote-2)
* 
* این مجموعه که زیر مجموعه ای از شرکت سامسونگ میباشد در زمینه‌های مختلفی سرویس ، آموزش و خدمات به Enterpriseها ارائه میکند. در زمینه‌ یادگیری ماشین به طور جدی از Kubeflow استفاده کرده و به صورت آموزش و در کنار شرکت های بزرگ همکاری میکند. در سایت این شرکت درباره سرویس AI خود به شرکت های بزرگ، Kubeflow را پلتفرم یادگیری ماشین مبتنی بر Kubernetes معرفی می کند.
* از جمله شرکت های بزرگ دیگر میتوان به Arrikto اشاره کرد. این شرکت مستقر در امریکا در واقع برای یادگیری ماشین Kubeflow as a Service ارائه میدهد.
* Arrikto، برخلاف سایر شرکت‌های MLOps، تیم‌های MLOps را قادر می‌سازد تا مدل‌های یادگیری ماشین را 30 برابر سریع‌تر از پلتفرم‌های سنتی ML به بازار عرضه کنند. Arrikto به عنوان یک مشارکت‌کننده پیشرو در Kubeflow، گردش‌های کاری خودکار، خطوط لوله قابل تکرار، استقرار مداوم از دسک‌تاپ به ابر و دسترسی ایمن به داده‌ها را ارائه می‌کند. Arrikto's Enterprise Kubeflow به عنوان یک توزیع چند گره در AWS، GCP و Azure در دسترس است و پلتفرم ترجیحی MLOps است که امروزه توسط بسیاری از شرکت های Fortune 500 در تولید استفاده می شود. این شرکت بیش از 300 مشتری در 17 کشور جهان دارد و توسط Unusual Ventures و Odyssey VP حمایت می شود.

# معماری و تکنولوژی‌های مرتبط

## معماری مفهومی

به‌طورکلی مراحل اصلی در معماری مفهومی MLOpsعبارت‌اند از:

1. **Data Preparation - آماده سازی داده‌ها**:

این مرحله شامل جمع‌آوری، تمیز کردن و پیش پردازش داده‌هایی است که برای آموزش و ارزیابی مدل‌های ML استفاده می‌شود. آماده‌سازی داده‌ها ممکن است شامل کارهایی مانند عادی‌سازی داده‌ها، تشخیص نقاط پرت و مهندسی ویژگی‌ها باشد.

1. **Model Selection - انتخاب مدل**:

در این مرحله، دانشمندان و مهندسان داده، معماری، الگوریتم و فراپارامترهای مدل ML مناسب را بر اساس ویژگی‌های داده‌ها و مسئله‌ای که باید حل شود، انتخاب می‌کنند.

**Model Training - آموزش مدل**:

این مرحله شامل استفاده از داده‌های آماده شده برای آموزش مدل انتخاب شده، با تنظیم پارامترهای مدل برای به حداقل رساندن خطا بین خروجی‌های پیش بینی شده و خروجی‌های واقعی است.

1. **Model Evaluation - ارزیابی مدل**:

این مرحله شامل ارزیابی عملکرد مدل آموزش‌دیده با مقایسه پیش‌بینی‌های آن با مجموعه‌ای از نتایج شناخته‌شده (که داده‌های آزمایشی نیز نامیده می‌شود)، و تعیین میزان تعمیم مدل به داده‌های نادیده جدید است.

1. **Model Deployment - استقرار مدل**:

این مرحله شامل در دسترس قرار دادن مدل برای کاربران نهایی است، خواه از طریق استقرار مدل در یک محیط تولید یا با ارائه یک API برای دسترسی دیگران به مدل باشد.

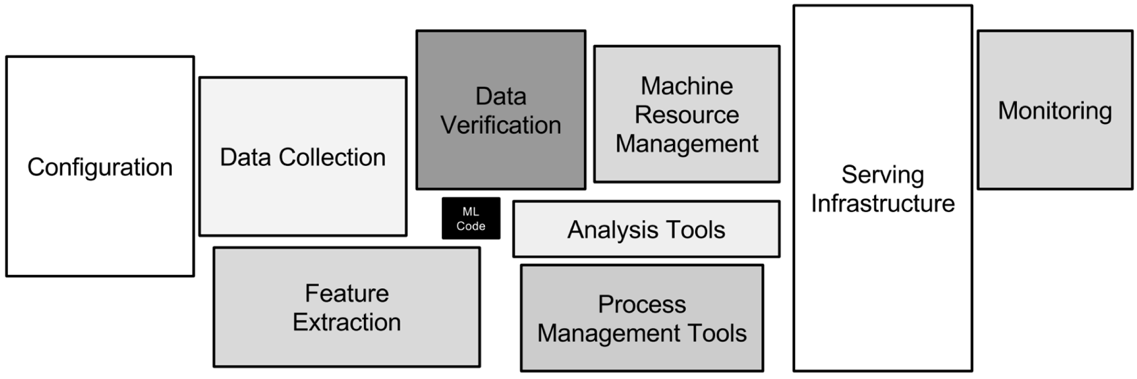
1. **Model Monitoring and Maintenance - نظارت و نگهداری مدل:**

این مرحله شامل نظارت بر عملکرد مدل مستقر در طول زمان و انجام به روز رسانی‌ها یا تنظیمات موردنیاز برای حفظ دقت و عملکرد مدل است.

**Model Retainment- بازنشستگی مدل:**

این مرحله شامل بازنشستگی مدل‌هایی است که دیگر موردنیاز نیستند یا دیگر عملکرد خوبی ندارند، تا اطمینان حاصل شود که فقط مدل‌هایی با عملکرد خوب به کار گرفته می‌شوند.

به‌طورکلی، هر یک از این مراحل نقش مهمی در چرخه زندگی یادگیری ماشین ایفا می‌کند و برای سازمان‌ها مهم است که یک فرآیند MLOps قوی و کارآمد را برای مدیریت چرخه عمر ML از ابتدا تا انتها اجرا کنند.



آماده سازی داده‌ها اولین مرحله در چرخه حیات یادگیری ماشینی (ML) است و شامل جمع‌آوری، تمیز کردن و پیش پردازش داده‌هایی است که برای آموزش و ارزیابی مدل‌های ML استفاده می‌شود. هدف از این مرحله آماده سازی داده‌ها در قالبی است که به راحتی توسط مدل ML مصرف شود و اطمینان حاصل شود که داده‌ها از کیفیت بالایی برخوردار هستند.

## مرحله آماده سازی داده‌ها در چرخه حیات ML

برخی از وظایف متداول درگیر در تهیه داده‌ها عبارتند از:

1. **Data Collection - جمع‌آوری داده‌ها**: جمع‌آوری داده‌ها از منابع مختلف مانند پایگاه‌های داده، فایل‌ها و APIهای خارجی.
2. **Data Cleaning - پاکسازی داده‌ها**: شناسایی و حذف هر گونه خطا، ناسازگاری یا موارد پرت در داده‌ها و اطمینان از کامل و دقیق بودن داده‌ها
3. **Data Integration - یکپارچه سازی داده‌ها:** ترکیب چندین منبع داده در یک مجموعه داده واحد، و حل هر گونه تضاد یا ناسازگاری که ممکن است ایجاد شود.
4. **Data Transformation - تبدیل داده‌ها:** تبدیل داده‌ها به قالبی که می‌تواند توسط مدل ML مصرف شود، مانند تبدیل متغیرهای طبقه بندی به مقادیر عددی.
5. **Data Normalization - عادی سازی داده‌ها**: مقیاس بندی داده‌ها به گونه ای که محدوده مقادیر یکسانی داشته باشند، که می‌تواند به بهبود عملکرد برخی از مدل‌های ML کمک کند.
6. **Feature Engineering - مهندسی ویژگی:** ایجاد ویژگی‌ها یا متغیرهای جدید که می‌تواند به بهبود عملکرد مدل ML کمک کند.
7. **Data Splitting - تقسیم داده‌ها:** تقسیم داده‌ها به مجموعه‌های آموزشی، اعتبار سنجی و تست، که برای آموزش، ارزیابی و آزمایش مدل ML استفاده می‌شود.

به‌طورکلی، آماده‌سازی داده‌ها مرحله‌ای حیاتی در چرخه حیات ML است، زیرا کیفیت و قالب داده‌ها به شدت بر دقت و عملکرد مدل نهایی ML تأثیر می‌گذارد. یک مجموعه داده به خوبی آماده شده می‌تواند به یک مدل دقیق و کارآمد منجر شود، و بنابراین، سرمایه‌گذاری زمان و منابع برای اطمینان از پاکسازی، تبدیل و آماده شدن داده‌ها برای مراحل بعدی چرخه حیات ML مهم است.

## معماری منطقی

معماری منطقی MLOps معمولاً شامل اجزای زیر است:

* مدیریت کد منبع: این مؤلفه کد منبع مدل‌های یادگیری ماشین، الگوریتم‌ها و خطوط لوله را ذخیره می‌کند.
* یکپارچه سازی مداوم/ استقرار مستمر (CI/CD): این جزء ادغام و استقرار مدل‌های یادگیری ماشین را خودکار می‌کند.
* آموزش مدل: این جزء با استفاده از داده‌ها و الگوریتم‌های ذخیره شده در جزء مدیریت کد منبع، مدل‌های یادگیری ماشین را آموزش می‌دهد.
* مدیریت مدل: این مؤلفه مدل‌های آموزش‌دیده و فراداده‌های مربوط به مدل‌ها مانند نسخه، دقت و داده‌های آموزشی را ذخیره می‌کند.
* استقرار مدل: این جزء، مدل‌های آموزش دیده را در یک محیط تولید مستقر می‌کند، جایی که می‌توان از آنها برای پیش بینی استفاده کرد.
* مانیتورینگ مدل: این مؤلفه عملکرد مدل‌های مستقر شده را نظارت می‌کند و در مورد دقت و قابلیت اطمینان آنها بازخورد ارائه می‌کند.
* مدیریت مدل: این مؤلفه چرخه عمر مدل‌های یادگیری ماشین را مدیریت می‌کند، از جمله نسخه‌سازی، به‌روزرسانی و بازنشستگی مدل‌ها در صورت لزوم.

## معماری فیزیکی

معماری فیزیکی MLOها به نیازهای خاص سازمان و پروژه‌های یادگیری ماشینی در حال توسعه بستگی دارد. زیرساخت باید مقیاس پذیر، انعطاف پذیر و قادر به رسیدگی به نیازهای جریان کار یادگیری ماشین و ادغام و استقرار مداوم مدل‌های یادگیری ماشین باشد. معماری فیزیکی MLOps به زیرساخت فیزیکی، سخت افزار و اجزای نرم افزاری مورد استفاده برای پیاده‌سازی معماری منطقی MLOها اشاره دارد. این می‌تواند شامل موارد زیر باشد:

* زیرساخت محاسباتی: این می‌تواند شامل سرورهای داخلی، ماشین‌های مجازی مبتنی بر ابر یا ترکیبی از هر دو باشد.
* ذخیره‌سازی داده‌ها: این می‌تواند شامل پایگاه‌های داده، دریاچه‌های داده یا سیستم‌های فایل مورد استفاده برای ذخیره داده‌های آموزشی، داده‌های آزمایشی و خروجی‌های مدل باشد.
* شبکه سازی: این می‌تواند شامل اجزای سخت افزاری و نرم افزاری باشد که از ارتباط بین اجزای مختلف معماری MLOps پشتیبانی می‌کند.
* نظارت: این می‌تواند شامل ابزارهای نظارت، تحلیلگرهای گزارش و داشبوردهایی باشد که برای نظارت بر عملکرد مدل‌های یادگیری ماشین و گردش کار MLOps استفاده می‌شوند.
* امنیت: این می‌تواند شامل فایروال‌ها، سیستم‌های کنترل دسترسی، رمزگذاری و سایر اقدامات امنیتی برای محافظت از داده‌ها و مدل‌های حساس درگیر در MLOها باشد.
* ابزارها و پلتفرم‌ها: این می‌تواند شامل ابزارها و پلتفرم‌های نرم‌افزاری منبع باز و تجاری باشد که برای یادگیری ماشین استفاده می‌شوند، مانند TensorFlow، PyTorch و غیره.

برخی ابزار/پلتفرم برتر MLOps جهت مدیریت چرخه یادگیری ماشین عبارتنداز:

* Amazon SageMaker
* Azure Machine Learning
* TensorFlow Extended (TFX)
* MLFlow
* Google Cloud ML Engine
* Data Version Control (DVC)
* H2O Driverless AI
* Kubeflow
* و ...

## بهترین ابزار منبع باز برای جمع‌آوری داده ها

* **Apache Nifi:** یک ابزار قدرتمند یکپارچه سازی داده ها است که به کاربران امکان می دهد به راحتی داده ها را از منابع مختلف جمع‌آوری، پردازش و مسیریابی کنند. این یک رابط کاربر پسند مبتنی بر وب دارد و از طیف گسترده ای از فرمت ها و پروتکل های داده پشتیبانی می کند.
* **Apache Kafka:** این یک پلتفرم پخش توزیع شده است که می تواند برای جمع‌آوری و پردازش حجم زیادی از داده ها در زمان واقعی استفاده شود. می‌تواند جریان‌های داده‌ای با توان عملیاتی بالا و تأخیر کم را مدیریت کند و دارای پشتیبانی داخلی برای تکرار داده‌ها و تحمل خطا است.
* **Apache Flume**: یک سرویس توزیع‌شده، قابل اعتماد و در دسترس برای جمع‌آوری، جمع‌آوری و جابجایی کارآمد مقادیر زیادی از داده‌های گزارش است. دارای معماری ساده و انعطاف پذیر است و طیف وسیعی از منابع و مقاصد داده را پشتیبانی می کند.
* **Scrapy**: این یک چارچوب اسکراپی وب پر استفاده است که می تواند برای استخراج داده ها از وب سایت ها و سایر منابع مبتنی بر وب استفاده شود. این در پایتون نوشته شده است و به کاربران اجازه می دهد تا عنکبوت های سفارشی را برای وب سایت ها و انواع داده های مختلف تعریف کنند.
* **Octoparse**: این یک ابزار خراش دادن وب است که می تواند برای استخراج داده ها از وب سایت ها و سایر منابع مبتنی بر وب استفاده شود. این یک رابط کاربر پسند نقطه و کلیک دارد و از طیف گسترده ای از فرمت ها و پروتکل های داده پشتیبانی می کند.
* **Beautiful Soup**: این یک کتابخانه پایتون برای تجزیه اسناد HTML و XML است که می تواند برای خراش دادن وب استفاده شود. درخت های تجزیه را از اسناد ایجاد می کند که می توانند برای استخراج داده ها از فایل های HTML و XML استفاده شوند.
* **Selenium**: این یک ابزار اتوماسیون مرورگر است که می تواند برای خراش دادن وب استفاده شود. این به کاربران اجازه می‌دهد مرورگرهای وب را خودکار کنند و می‌توان از آن برای حذف داده‌ها از وب‌سایت‌هایی که توسط جاوا اسکریپت ارائه می‌شوند استفاده کرد.
* به‌طورکلی، این ابزارها را می‌توان برای جمع‌آوری داده‌ها از منابع مختلف و در قالب‌های مختلف، بسته به نیاز پروژه و نیازهای خاص فرآیند جمع‌آوری داده‌ها، مورد استفاده قرار داد.

## ذخیره سازی داده‌ها

بسته به نیازهای خاص پروژه و اندازه و پیچیدگی داده‌ها، گزینه‌های مختلفی برای ذخیره داده‌ها وجود دارد. برخی از گزینه‌های رایج عبارتند از:

* **پایگاه داده‌های رابطه ای SQL DATABASES**: این ها پایگاه‌های داده سنتی هستند که از ساختار جدولی برای ذخیره داده‌ها استفاده می‌کنند. پایگاه داده‌های رابطه ای محبوب عبارتند از MySQL، PostgreSQL و Microsoft SQL Server. آنها به خوبی برای داده‌های ساختاریافته مناسب هستند و قابلیت‌های جستجو و نمایه سازی قدرتمندی را ارائه می‌دهند.
* **پایگاه‌های داده NoSQL:** اینها پایگاه‌های اطلاعاتی غیر رابطه‌ای هستند که از یک ساختار سند، کلید-مقدار یا ساختار مبتنی بر نمودار برای ذخیره داده‌ها استفاده می‌کنند. پایگاه داده‌های معروف NoSQL شامل MongoDB، Cassandra و Neo4j می‌باشد. آنها به خوبی برای داده‌های بدون ساختار و نیمه ساختار یافته مناسب هستند و اغلب برای ذخیره مقادیر زیادی از داده‌ها استفاده می‌شوند.
* **ذخیره‌سازی داده‌های مبتنی بر ابر:** اینها راه‌حل‌های ذخیره‌سازی داده‌هایی هستند که بر روی پلتفرم‌های ابری مانند خدمات وب آمازون (AWS)، مایکروسافت آژور و پلتفرم ابری Google (GCP) میزبانی می‌شوند. آنها مقیاس پذیری، در دسترس بودن بالا، و پشتیبان گیری داخلی داده‌ها و ویژگی‌های بازیابی فاجعه را ارائه می‌دهند.

**انبارهای داده:** برای ذخیره و مدیریت مقادیر زیادی از داده‌ها طراحی شده اند که معمولاً برای تجزیه و تحلیل و گزارش استفاده می‌شوند. داده‌ها را می‌توان بارگذاری کرد و به قالبی تبدیل کرد که برای پرس و جو و تجزیه و تحلیل بهینه شده است.

**سیستم‌های فایل:** برای ذخیره داده‌ها در سیستم‌های فایل محلی یا راه دور مانند HDFS یا GlusterFS استفاده می‌شوند. آنها را می‌توان برای ذخیره مقادیر زیادی داده استفاده کرد و اغلب همراه با راه‌حل‌های دیگر ذخیره‌سازی داده مانند انبارهای داده یا ذخیره‌سازی داده مبتنی بر ابر استفاده می‌شود.

**سیستم ذخیر سازی شی:** در سیستم‌های مدرن استفاده از Object Storage ها مانند S3 می‌تواند یکی از گزینه‌های بسیار مناسب برای ذخیره‌سازی انواع دیتا و فایل نیز باشد. از جمله ابزار معروف آن می‌توان به MinIO اشاره نمود.

در نهایت انتخاب راه حل ذخیره‌سازی به نیازهای خاص پروژه، مانند اندازه و پیچیدگی داده‌ها، عملکرد و مقیاس پذیری موردنیاز، و بودجه و منابع موجود بستگی دارد. ترکیبی از راه حل‌های مختلف ذخیره‌سازی نیز بسته به مورد استفاده می‌تواند مورد استفاده قرار گیرد. در مرحله آماده سازی داده‌های ML، می‌توانیم داده‌های خام جمع‌آوری شده را در دریاچه‌های داده و انبارهای داده ذخیره کنیم. ذخیره داده‌های خام جمع‌آوری‌شده در دریاچه‌های داده و انبارهای داده در مرحله آماده‌سازی داده‌های چرخه حیات ML معمول است. هم دریاچه‌های داده و هم انبارهای داده برای ذخیره مقادیر زیادی از داده‌ها و ارائه قابلیت‌های جستجو و نمایه سازی قدرتمند طراحی شده اند که می‌تواند برای کارهای آماده سازی داده‌ها مانند تمیز کردن داده‌ها، تبدیل و مهندسی ویژگی‌ها مفید باشد. برخی از گزینه‌های منبع باز محبوب برای دریاچه‌های داده و انبارهای داده عبارتند از:

**دریاچه‌های داده[[3]](#footnote-3)**

**Apache Hadoop**

یک چارچوب منبع باز محبوب برای ذخیره‌سازی توزیع شده و پردازش مجموعه داده‌های بزرگ. این شامل سیستم فایل HDFS و مدل برنامه نویسی MapReduce است.

ابزارهای مورد استفاده همراه با Hadoop

**Apache Hive**

انبار داده و زبان جستجوی SQL مانند برای Hadoop.

**Apache Spark**

یک سیستم محاسباتی خوشه ای سریع و همه منظوره برای پردازش کلان داده‌ها. این می‌تواند داده‌های ذخیره شده در HDFS و سایر سیستم‌های ذخیره‌سازی را پردازش کند.

**Apache Nifi**

یک سیستم آسان برای استفاده، قدرتمند و قابل اعتماد برای پردازش و توزیع داده‌ها.

**S3 Object Storage**

Amazon S3 (سرویس ذخیره‌سازی ساده) می‌تواند به عنوان دریاچه داده استفاده شود. S3 یک سرویس ذخیره‌سازی اشیاء مبتنی بر ابر است که مقیاس پذیری نامحدود و دوام بالا را ارائه می‌دهد و آن را به گزینه ای مناسب برای ذخیره حجم زیادی از داده‌های خام تبدیل می‌کند. داده‌های ذخیره‌شده در S3 را می‌توان به راحتی توسط ابزارهای پردازش و تحلیل داده‌های مختلف، مانند Amazon EMR (Elastic MapReduce)، که سرویسی است که به شما امکان می‌دهد کارهای پردازش کلان داده‌های توزیع‌شده را بر روی مجموعه‌ای از نمونه‌های EC2 اجرا و پردازش کرد، قابل دسترسی و پردازش است. چسب AWS، که یک سرویس استخراج، تبدیل و بارگذاری (ETL) کاملاً مدیریت شده است که می‌تواند برای آماده سازی و پردازش داده‌های ذخیره شده در S3 استفاده شود.

S3 همچنین به شما امکان می‌دهد از ویژگی‌هایی مانند نسخه‌سازی، سیاست‌های چرخه حیات و تکرار بین منطقه‌ای استفاده کنید، که می‌تواند برای اطمینان از دوام داده‌ها و حاکمیت داده‌ها استفاده شود، این یک ویژگی کلیدی برای استفاده از دریاچه داده است.

Amazon S3 همچنین می‌تواند با سایر سرویس‌های AWS، مانند Amazon Athena، که یک سرویس پرس و جو بدون سرور و تعاملی است که به شما امکان تجزیه و تحلیل داده‌های ذخیره شده در S3 با استفاده از SQL را می‌دهد، و Amazon Redshift، که یک سرویس انبار داده کاملاً مدیریت شده است، ادغام شود. برای ذخیره، پرس و جو و تجزیه و تحلیل حجم زیادی از داده‌ها استفاده شود.

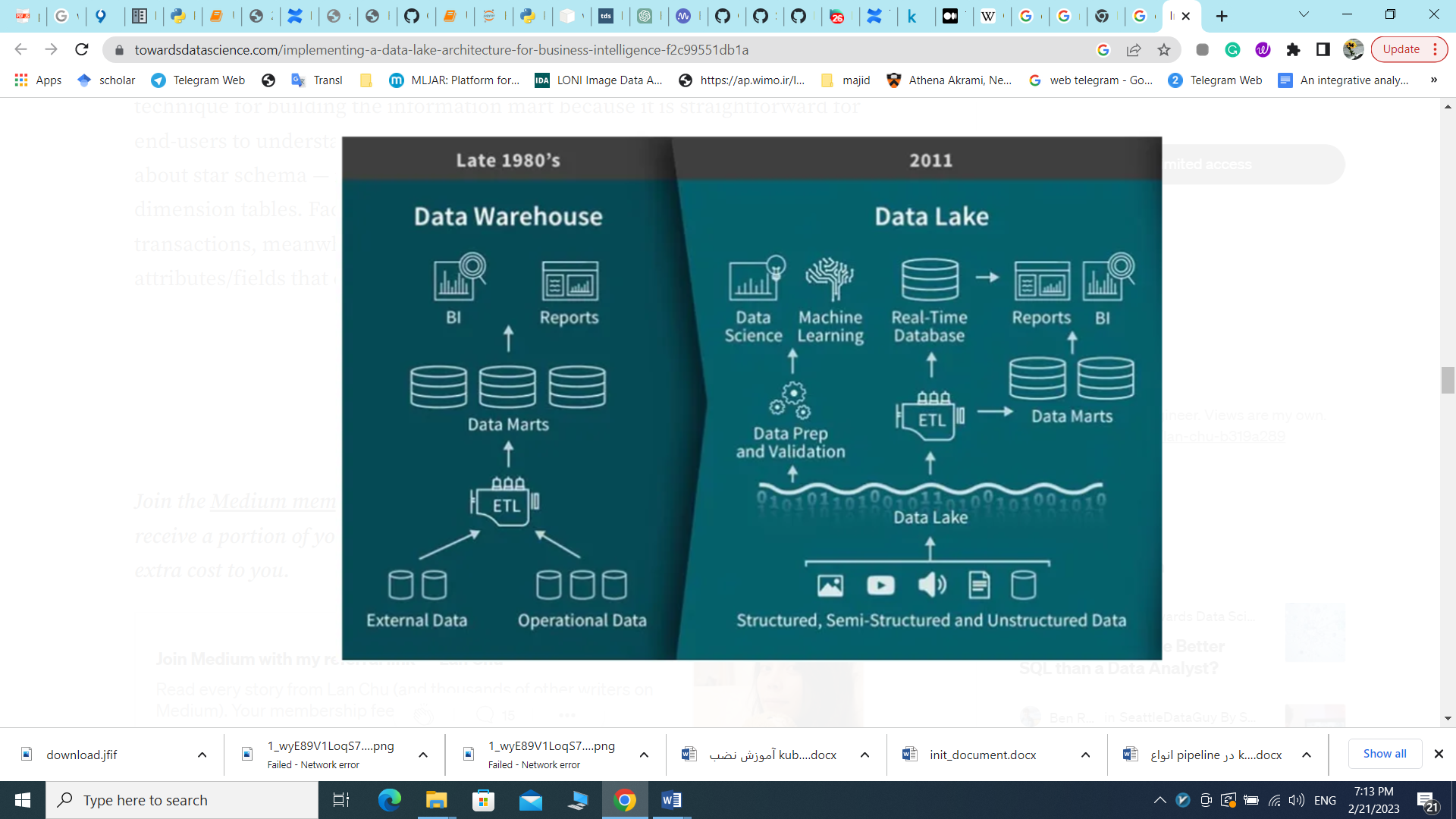
شایان ذکر است که S3 یک سرویس ذخیره‌سازی است و یک پلت فرم مدیریت دریاچه داده نیست و برای پیاده‌سازی کشف داده‌ها، مدیریت داده‌ها، مدیریت داده‌ها، نسل داده‌ها و غیره باید از ابزارها یا خدمات دیگری استفاده کنید. اما S3 می‌تواند به عنوان پایه ای برای ساخت دریاچه داده با سایر خدمات AWS.

**انبارهای داده[[4]](#footnote-4)**

در محاسبات، انبار داده، همچنین به عنوان انبار داده سازمانی شناخته می‌شود، سیستمی است که برای گزارش‌گیری و تجزیه و تحلیل داده‌ها استفاده می‌شود و جزء اصلی هوش تجاری در نظر گرفته می‌شود. DWها مخازن مرکزی داده‌های یکپارچه از یک یا چند منبع متفاوت هستند.

* **Amazon RedShift**
* **Amazon DynamoDB**
* **Oracle Database**
* **Microsoft Azure Synapse**
* **IBM DB2**
* **Terradata**
* **SAP**
* **Google BigQuery**
* **Snowflake**
* **Cloudera**
* **Micro Focus**
* **PostgreSQL**
* **MariaDB**
* **MongoDB**
* **CockroachDB**
* **InfluxDB**
* **Firebolt**
* **Talend Open Studio**
* **TabLeau**
* **Pentaho**
* **Panoply**
* **Micro Focus Vertica**
* **Amazon S3**
* **Microsoft SQL Server**
* **Databricks Lakehouse Platform**
* **Neo4J**
* **…**

همانطور که ملاحظه می‌کنید ابزارها و دیتابیس‌ها و گزینه‌های زیادی برای انبار داده می‌توان استفاده کرد. که به نوع دیتا، زیرساخت در اختیار و عوامل مختلفی می‌تواند بستگی داشته، همچنین ممکن است در یک زیرساخت یادگیری ماشین کامل از چندین گزینه همزمان استفاده شود.



## بهترین دریاچه داده برای MLOps

بهترین دریاچه داده برای MLOها به نیازهای خاص سازمان شما و مورد استفاده ای که روی آن کار می‌کنید بستگی دارد. برخی از گزینه‌های منبع باز محبوب برای دریاچه‌های داده عبارتند از:

**Apache Hadoop: Hadoop**

یک چارچوب پردازش و ذخیره‌سازی داده منبع باز است که معمولاً برای تجزیه و تحلیل داده‌های بزرگ استفاده می‌شود. این شامل یک سیستم فایل توزیع شده (HDFS) و یک چارچوب پردازش داده توزیع شده (MapReduce) است که می‌تواند برای ذخیره و پردازش مقادیر زیادی داده استفاده شود.

**Apache Nifi**

یک ابزار قدرتمند برای یکپارچه سازی داده‌ها و مدیریت داده‌ها است که جمع‌آوری کارآمد، مدیریت و مدیریت داده‌ها را امکان پذیر می‌کند. می‌توان از آن برای خودکارسازی وظایف جمع‌آوری و پیش‌پردازش داده‌ها استفاده کرد و همچنین می‌توان از آن برای پیاده‌سازی سلسله داده‌ها و مدیریت ابرداده استفاده کرد.

**Apache Kylin**

این یک انبار داده تحلیلی منبع باز و توزیع شده است که دسترسی شبیه به SQL را به داده‌های ذخیره شده در Hadoop فراهم می‌کند و پرس و جوهای OLAP با تاخیر کم را در داده‌های بزرگ فعال می‌کند.

**Apache Druid**

این یک فروشگاه داده ستونی با کارایی بالا است که برای پرس و جوهای OLAP در مجموعه‌های داده در مقیاس بزرگ طراحی شده است. می‌توان از آن برای ذخیره و پرس و جو مقادیر زیادی از داده‌های سری زمانی استفاده کرد که آن را برای موارد استفاده که شامل داده‌های سری زمانی می‌شود مناسب می‌کند.

**MinIO**

همانطور که قبلاً اشاره کردم MinIO یک سرور ذخیره‌سازی شی منبع باز است که برای سازگاری با Amazon S3 طراحی شده است. می‌توان از آن به عنوان یک دریاچه داده برای پلت فرم MLOps استفاده کرد زیرا بسیاری از ویژگی‌های مشابه S3 را ارائه می‌دهد، از جمله مقیاس پذیری، دوام و سازگاری با طیف گسترده ای از پردازش داده‌ها و ابزارهای تجزیه و تحلیل.

در نهایت، بهترین دریاچه داده برای ML به نیازهای خاص سازمان شما بستگی دارد، مانند انواع داده‌هایی که باید ذخیره کنید، اندازه مجموعه داده‌های شما، الزامات عملکرد و مقیاس‌پذیری، و ابزارها و فناوری‌های خاصی که دارید. MinIO در درجه اول یک سرویس ذخیره‌سازی است و عملکرد داخلی را برای نسل داده‌های خارج از جعبه ارائه نمی‌دهد. اصل و نسب داده توانایی ردیابی منشاء و تکامل داده‌ها است، از جمله اینکه چگونه ایجاد شده اند، از کجا آمده اند، چگونه تبدیل شده اند و به کجا ختم شده اند.

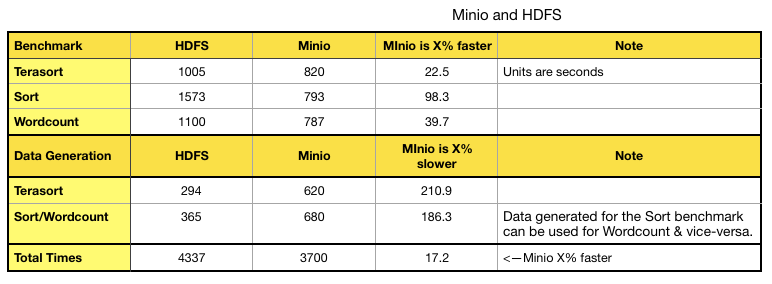
اصل و نسب داده در MLOها مهم است زیرا به شما امکان می‌دهد نحوه استفاده از داده‌ها، نحوه تبدیل و نحوه استفاده از آن برای آموزش مدل را درک کنید. این موضوع در ممیزی و عیب یابی مسائل در خط لوله ML شما مهم است.

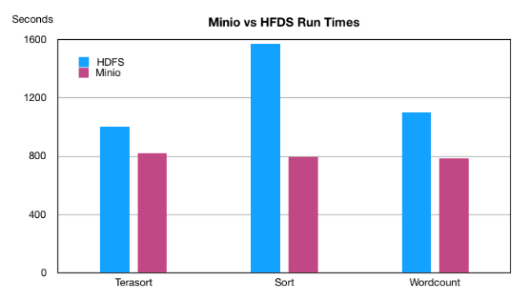
در حالی که MinIO عملکرد خط داده را ارائه نمی‌دهد، می‌تواند با ابزارها یا خدمات دیگری که چنین عملکردی را ارائه می‌دهند یکپارچه شود. به عنوان مثال، می‌توانید از Apache Atlas، که یک راه‌حل مدیریت داده و مدیریت ابرداده منبع باز است، برای مدیریت سلسله داده‌ها و ابرداده‌ها برای داده‌های ذخیره شده در MinIO استفاده کنید. جایگزین‌های دیگر برای خط داده‌ها Apache Nifi، Apache Atlas یا راه حل‌های تجاری مانند Informatica، Collibra یا Alation هستند.

علاوه بر این، MinIO API های ساده ای را برای مدیریت ابرداده ارائه می‌دهد و می‌تواند با ابزارها یا خدمات دیگری که چنین عملکردی را برای دستیابی به خط داده‌ها ارائه می‌دهند، ادغام شود.

**مقایسه عملکرد Hadoop و MinIO**

خانه نرم‌افزار ذخیره‌سازی اشیاء MinIO نشان داده است که ذخیره‌سازی آن می‌تواند تا ۹۳ درصد سریع‌تر از سیستم Hadoop اجرا شود. در آخرین معیارهای خود که این هفته منتشر شد، MinIO سریع‌تر از پیکربندی سیستم فایل Hadoop (HDFS) بود. در راه اندازی آزمایشی، هر دو سیستم در ابر عمومی آمازون اجرا می‌شدند.





بهترین نتیجه MinIO در مرحله Sort 93 درصد سریع‌تر بود. با جمع‌بندی زمان‌های تولید داده‌های MinIO و HDFS به زمان‌های اجرای آزمایشی، می‌بینیم که MinIO (3700 ثانیه) در کل سریع‌تر از HDFS (4337 ثانیه) بود. می‌توانید جزئیات معیار Minio HDFS را بررسی کنید. ماه گذشته این شرکت معیارهایی را منتشر کرد که نشان می‌داد نرم افزار منبع باز سازگار با S3 سریع‌تر از Presto و Apache Spark است.

<https://blocksandfiles.com/2019/08/16/minio-object-storage-hadoop-benchmarks/#:~:text=Object%20storage%20software%20house%20MinIO,in%20the%20Amazon%20public%20cloud>

<https://blog.min.io/hdfsbenchmark//>

**Airbyte ابزاری در فرایند جمع‌آوری دیتا**

Airbyte یک پلت فرم یکپارچه سازی داده منبع باز است که می‌تواند برای جمع‌آوری داده‌ها، در میان سایر وظایف یکپارچه سازی داده‌ها استفاده شود. این به کاربران اجازه می‌دهد تا به راحتی داده‌ها را از منابع مختلف مانند پایگاه‌های داده، برنامه‌های SaaS و ذخیره‌سازی ابری به یک مکان متمرکز جمع‌آوری، تکثیر و انتقال دهند. برخی از ویژگی‌های Airbyte عبارتند از:

پشتیبانی از طیف گسترده ای از منابع و مقاصد داده، از جمله PostgreSQL، MySQL، Amazon S3 و Google BigQuery.

* تکثیر بلادرنگ داده و بارگذاری افزایشی، که به کاربران امکان می‌دهد داده‌های خود را به روز نگه دارند.
* یک رابط مبتنی بر وب ساده و کاربردی برای پیکربندی و مدیریت خطوط لوله داده.
* یک معماری مقیاس پذیر و مقاوم در برابر خطا که می‌تواند حجم زیادی از داده‌ها را مدیریت کند.
* پشتیبانی داخلی برای بررسی کیفیت داده‌ها، که می‌تواند به اطمینان از دقیق و کامل بودن داده‌های جمع‌آوری شده کمک کند.
* یک API مناسب برای توسعه‌دهندگان که به کاربران اجازه می‌دهد برای سایر منابع داده و مقاصد اتصالات سفارشی ایجاد کنند.   
  Airbyte نسبتاً جدید است و در فضای یکپارچه سازی داده‌ها محبوبیت پیدا می‌کند، ارزش آن را دارد که آن را به عنوان یک گزینه در نظر بگیرید.

# ویژگی‌های کارکردی

## شکاف‌های ارتباطی تیم‌های تجاری و فنی

چالش اصلی بین تیم‌های تجاری و فنی ارتباط است. پیدا کردن زبان مشترک جهت همکاری سخت است. بیشتر اوقات این شکاف دلیل شکست پروژه‌های بزرگ است.

## ارزیابی ریسک

ارزیابی ریسک نیز یکی از چالش‌های مهم است، زیرا مدل‌ها اغلب تمایل دارند از آنچه در ابتدا نتایج می‌دادند دور شوند. ارزیابی هزینه/خطر چنین شکست‌هایی یک کار ضروری و چالش برانگیز است.

## بازتاب تغییر اهداف کسب و کار در مدل

وابستگی‌های زیادی با داده‌ها وجود دارد که به طور مداوم عملکرد را حفظ می‌کند، استانداردهای مدل را تغییر می‌دهد و از حاکمیت هوش مصنوعی اطمینان می‌دهد. همگام شدن با اهداف در حال تحول کسب و کار و آموزش مدل مستمر چالش برانگیز است.

# ویژگی‌های غیرکارکردی

## آمادگی hybrid-cloud

## امنیت

## مقیاس پذیری

## تحمل خطا

## ارائه توان عملیاتی بالا

## پاسخ سریع

# فرضیات و محدودیت‌ها

MLOps جهت استقرار مؤثر مدل ML در دنیای واقعی ضروری است، اما هزینه آن به دلیل پیچیدگی و منابع موردنیاز می‌تواند بسیار گران باشد. عامل اصلی تعیین‌کننده در استفاده یا عدم استفاده از MLOps ها، بررسی کامل بازگشت سرمایه با بگارگیری یادگیری ماشین است.

* پیاده‌سازی MLOpsها ممکن است گران باشد.
* توانایی ایجاد و حفظ یک چرخه حیات MLOps به طور مؤثر به منابع کافی (افراد، زیرساخت‌ها و ابزار) نیاز دارد.
* MLOps اگر به درستی انجام نشود، به ابزارها و مهارت‌های فراوانی نیاز دارد که بر پیچیدگی آن افزوده می‌شود.
* جهت ساختن یک چارچوب موفق MLOps ، به یک معمار باتجربه MLOps و مهندسان ML نیاز داریم که از قبل با اصول مفهوم DevOps آشنا هستند.

# تست و ارزیابی MLOPs

پس از ایجاد تست‌ها، می‌توانیم هر زمان که تغییری در سیستم ایجاد شد، تست‌ها را به‌طور خودکار اجرا شود و در طول زمان به بهبود آنها ادامه دهیم. جهت جلوگیری از افزایش هزینه‌های پایین دستی و از دست دادن زمان، پاداش دادن به اجرای تست‌ها و شناسایی منابع اشتباه در سریع‌ترین زمان ممکن در چرخه توسعه، مناسب خواهد بود.

## تست مدل

در این مرحله، عملکرد مدل آموزش‌دیده را بر روی مجموعه‌ای از نقاط داده مجزا به نام داده‌های آزمایشی (که در مرحله دریافت داده تقسیم و نسخه‌سازی شد) ارزیابی می‌شود. استنباط مدل آموزش دیده بر اساس معیارهای انتخاب شده بر اساس مورد استفاده ارزیابی می‌شود. خروجی این مرحله گزارشی از عملکرد مدل آموزش دیده است. در ماژول Deploy، مدل‌های آموزش دیده را به ترتیب در محیط‌های توسعه‌دهنده، تست و تولید مستقر می‌شود. ابتدا با تست اپلیکیشن (که در محیط‌های توسعه دهنده و تست انجام می‌شود) شروع می‌شود.

## تست برنامه

قبل از استقرار یک مدل یادگیری ماشینی جهت تولید، تست استحکام، مقیاس پذیری و امنیت مدل بسیار مهم است. از این رو، ما مرحله "تست برنامه" را داریم که در آن تمام مدل‌های آموزش دیده و برنامه کاربردی را در یک محیط تولید مانند به نام محیط تست یا مرحله بندی به طور دقیق تست می‌شود. در این مرحله، ما ممکن است تست‌هایی مانند تست‌های A/B، تست‌های یکپارچه سازی، تست‌های پذیرش کاربر (UAT) ، یا تست بار را انجام دهیم.

# آموزش و تحویل نصب و راه اندازی MLOPs

## آموزش نصب Kubeflow

Kubeflow مجموعه ای از ابزارهای لوکال ابری (معمولاً به عنوان کامپوننت) برای توسعه و نگهداری تمام مراحل MDLC است. Kubeflow بر روی ابزار ارکستراسیون کانتینر Kubernetes ساخته شده است. پیکربندی و هماهنگ کردن هر مرحله از MDLC به طور مستقیم در یک خوشه Kubernetes می تواند یک فرآیند پیچیده و زمان بر برای مهندسان یادگیری ماشین باشد. بنابراین، Kubeflow پلتفرمی را ارائه می دهد که ابزارهایی را برای پیکربندی، توسعه، خودکارسازی و استقرار هر مرحله از MDLC در یک خوشه Kubernetes فراهم می کند. این باعث می شود پیکربندی و اعمال تغییرات به طور مستقیم در یک خوشه Kubernetes آسان تر و زمان کمتری صرف شود. چرخه عمر توسعه مدل (MDLC) اصطلاحی است که معمولاً برای توصیف جریان بین آموزش مدل و استنتاج استفاده می شود. آموزش به فرآیند ایجاد مدل یادگیری ماشین و استنتاج به فرآیند استفاده از یک مدل یادگیری ماشین آموزش دیده برای پیش بینی اشاره دارد.

مراحل MDLC را می توان به موارد زیر تقسیم کرد:

data exploration -> feature preparation -> model training and model tuning -> model serving -> model testing -> model versioning.

## راه اندازی Kubeflow

برای استقرار موفقیت آمیز Kubeflow، می بایست یک کلاستر Kubernetes داشته باشیم. از آنجایی که خود Kubernetes به منابع زیادی نیاز دارد، ما از نرم‌افزاری استفاده می‌کنیم که کلاستر Kubernetes را شبیه‌سازی می‌کند، مانند Minikube، Kind، K3 و … . ما از Minikube برای این مورد استفاده خواهیم کرد.

۱- نصب Docker:

apt-get update

apt-get install -y apt-transport-https ca-certificates curl software-properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add -

add-apt-repository \"deb [arch=amd64] https://download.docker.com/linux/ubuntu \$(lsb\_release -cs) \stable"

apt-get update

apt-get install docker-ce docker-ce-cli containerd.io

قبل از نصب Kubectl باید swap را غیرفعال کنیم، در غیر این صورت Kubectl به درستی کار نخواهد کرد:

$ sudo swapoff -a

۲- نصب kubectl:

$ curl -LO [https://storage.googleapis.com/kubernetes](https://storage.googleapis.com/kubernetes-release/release/v1.15.0/bin/linux/amd64/kubectl)

[release/release/v1.15.0/bin/linux/amd64/kubectl](https://storage.googleapis.com/kubernetes-release/release/v1.15.0/bin/linux/amd64/kubectl)

$ chmod +x ./kubectl

$ mv ./kubectl /usr/local/bin/kubectl

۳- نصب Kubernetes:

به کمک دستور زیر Minikube را نصب می کنیم.

$ minikube start -- name kubeflow --kubernetes-version=v1.20.1

ما نسخه kubernetes را مشخص می کنیم زیرا کامپوننت apiextensions.k8s.io/v1beata1 در kubernetes نسخه 1.22.0 و جدیدتر منسوخ شده است. ما همچنین نام کلاستر را در صورتی که بخواهیم کلاسترهای مختلفی را به طور همزمان ایجاد کنیم، مشخص می کنیم، تا به ما امکان آن را دهد تا کلاستر مورد نظر را به دلخواه شروع و متوقف کنیم. برای تأیید اینکه کلاستر ما آماده و در حال اجرا است، دستور زیر را اجرا می کنیم:

$ kubectl cluster-info

برای نصب Kubeflow ابتدا باید آبجکت kfctl را نصب کنیم، kfctl امکان کنترل برای استقرار و مدیریت Kubeflow را به ما می دهد. براش شروع ابتدا متغیرهای محیطی را تعریف می کنیم:

$ PLATFORM=$(uname)

$ export PLATFORM

$ mkdir -p ~/Kubeflow/bin

$ export KUBEFLOW\_TAG=1.2.0

$ KUBEFLOW\_BASE="https://api.github.com/repos/kubeflow/kfctl/releases"

$ KFCTL\_URL=$(curl -s ${KUBEFLOW\_BASE} | grep http | grep "${KUBEFLOW\_TAG}" | grep -i "${PLATFORM}" | cut -d : -f 2,3 | tr -d '\" ' )

بلاک دستورات بالا متغیرهای PLATFORM، KUBEFLOW\_TAG، KUBEFLOW\_BASE و KFCTL\_URL را ایجاد می کند که نشان دهنده پلتفرم فعلی است که ما دستورات را روی آن اجرا می کنیم (لینوکس)، نسخه kfctl که می خواهیم، ​​URL به همه نسخه های kfctl و URL نسخه kfctl که ما مشخص کرده ایم. همچنین دایرکتوری ~/Kubeflow/bin را برای ذخیره باینری های kubectl ایجاد می کنیم.

$ wget "${KFCTL\_URL}"

$ KFCTL\_FILE=${KFCTL\_URL##\*/}

$ tar -xvf "${KFCTL\_FILE}"

$ mv ./kfctl ~/Kubeflow/bin

از ابزار wget برای دانلود tar فایل kfctl استفاده می کنیم، نام فایل tar دانلود شده را در بلوک فرمان قبلی دریافت می کنیم و آن را در متغیر KFCTL\_FILE ذخیره می کنیم و سپس با استفاده از دستور tar آن را استخراج می کنیم. سپس محتویات فایل tar (برنامه kfctl) را به دایرکتوری که قبلا ایجاد کردیم منتقل می کنیم.

دایرکتوری (kfctl (~/Kubeflow/bin را به متغیر PATH اضافه می کنیم تا از آن در ترمینال استفاده کنیم.

$ export PATH=$PATH:~/Kubeflow/bin

توجه: دستور فوق این اجازه را می دهد تا kfctl فقط در ترمینال جاری استفاده شود، پس از بستن ترمینال باید دوباره دستور را اجرا کنید. برای افزودن دائمی آن، باید بسته به پوسته ای که استفاده می کنید، دستور را به ~/.zshrc یا ~/.bashrc اضافه کنید.

برای بررسی صحت نصب kfctl، دستور زیر را اجرا می کنیم:

$ kfctl version

Docker و Kubernetes از فایل‌های yaml برای پیکربندی استفاده می‌کنند، بنابراین جای تعجب نیست که Kubeflow از فایل‌های yaml نیز استفاده می‌کند یا در تهیه context manifestها از آنها نام برده می شود. فایل های مانیفست تمام سرویس های kubeflow (کامپوننت ها) را برای استقرار در این کلاستر کوبرنتیز تعریف می کنند. همانطور که قبل تر انجام دادیم، می بایست متغیرهای محیطی را تعریف کنیم:

$ MANIFEST\_BRANCH=${MANIFEST\_BRANCH:-v1.2-branch}

$ export MANIFEST\_BRANCH

$ MANIFEST\_VERSION=${MANIFEST\_VERSION:-v1.2.0}

$ export MANIFEST\_VERSION

متغیر MANIFEST\_BRANCH مشخص می کند که کدام نسخه از Kubeflow از شاخه GitHub که در آن نسخه Kubeflow قرار دارد و متغیر MANIFEST\_VERSION نسخه را در فایل های مانیفست ما مشخص می کند.

توجه: می توانید ریپو مانیفست kubeflow را بررسی کنید تا از آخرین نسخه در دسترس اطلاع پیدا کنید. به عنوان مثال برای دیدن همه نسخه های موجود در شاخه v1.2 می توانید به اینجا مراجعه کنید.

https://github.com/kubeflow/manifests/tree/v1.2 -branch/kfdef

$ KF\_PROJECT\_NAME=${KF\_PROJECT\_NAME:-adin-kf-${PLATFORM}}

$ export KF\_PROJECT\_NAME

$ mkdir "${KF\_PROJECT\_NAME}"

ما نام پروژه Kubelfow را تعریف می کنیم، در این صورت adin-kf- {پلتفرمی که در آن هستید} خواهد بود. سپس یک دایرکتوری (با همان نام با نام پروژه) ایجاد می کنیم که در آن فایل های مانیفست را ذخیره می کنیم.

$ manifest\_root=https://raw.githubusercontent.com/kubeflow/manifests

$ FILE\_NAME=kfctl\_k8s\_adin.${MANIFEST\_VERSION}.yaml

$ KFDEF=${manifest\_root}${MANIFEST\_BRANCH}/kfdef/${FILE\_NAME}

ما متغیر manifest\_root را تعریف می‌کنیم که شامل URLی است که به فایل‌های raw در ریپو kubeflow/manifests اشاره می‌کند، متغیر FILE\_NAME حاوی نام فایل مانیفست مورد نظر ما و متغیر KFDEF است که متغیرهای قبلاً تعریف‌شده را ترکیب می‌کند تا URLی را ایجاد کند که به فایل مانیفست که مورد نظر ما اشاره می کند.

$ kfctl apply -f $KFDEF -V

برای دانلود و اعمال فایل مانیفست دستور بالا را اجرا می کنیم. پیام‌های وضعیت، برخی از پیام‌های هشدار و گاهی اوقات حتی پیام‌های خطا را مشاهده خواهید کرد، اما اگر دستور زیر 0 را برگرداند، یعنی می بایست ادامه دهید.

$ echo $?

توجه: تکمیل فرآیند استقرار ممکن است تا 30 دقیقه طول بکشد.

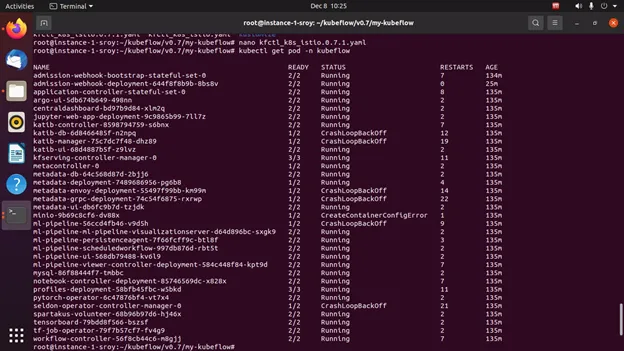
بررسی وضعیت:

برای اطمینان از موفقیت آمیز بودن نصب، دستور زیر را اجرا می کنیم. اگر همه POD ها در حالت RUNNING یا COMPLETED باشند، استقرار موفقیت آمیز بود.

$ kubectl get pods --all-namespaces -w

توجه: اگر PODی در حالت دیگری به جز running یا complete دارید، می توانید دستور زیر را اجرا کنید تا اطلاعات بیشتری در مورد مشکل پیدا کنید:

$ kubectl describe pod [pod-name] –namespace knative-serving



## داشبورد Kubeflow

از آنجایی که کامپوننت های ما به صورت لوکال اجرا می شوند، برای دسترسی به Kubeflow UI تنها کاری که باید انجام دهیم یک پورت فوروارد ساده است:

$ kubectl port-forward svc/adin-ingressgateway -n adin-system 12014:80

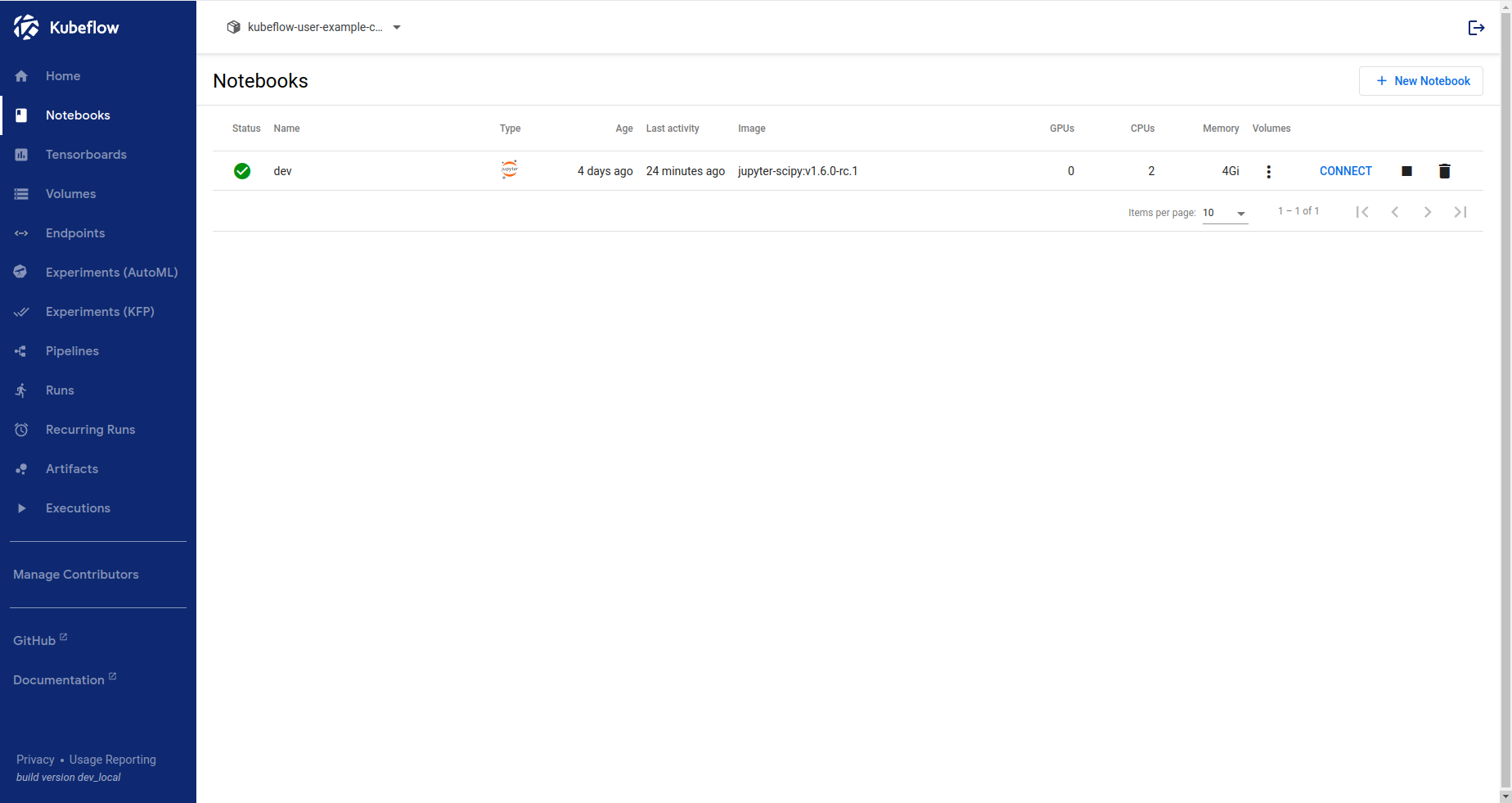
ما از Kubectl استفاده می کنیم تا پورت 80 پاد adin-ingressgateway را از namespaceی به نام adin-system مشاهده کنیم و داده هایی را که از این پورت عبور می کنند را به پورت 12014 ماشین localhost ارسال می کنیم. در نهایت برای دسترسی به داشبورد، http://localhost:12014 را در مرورگر خود وارد می کنیم.

**به نام خدا**

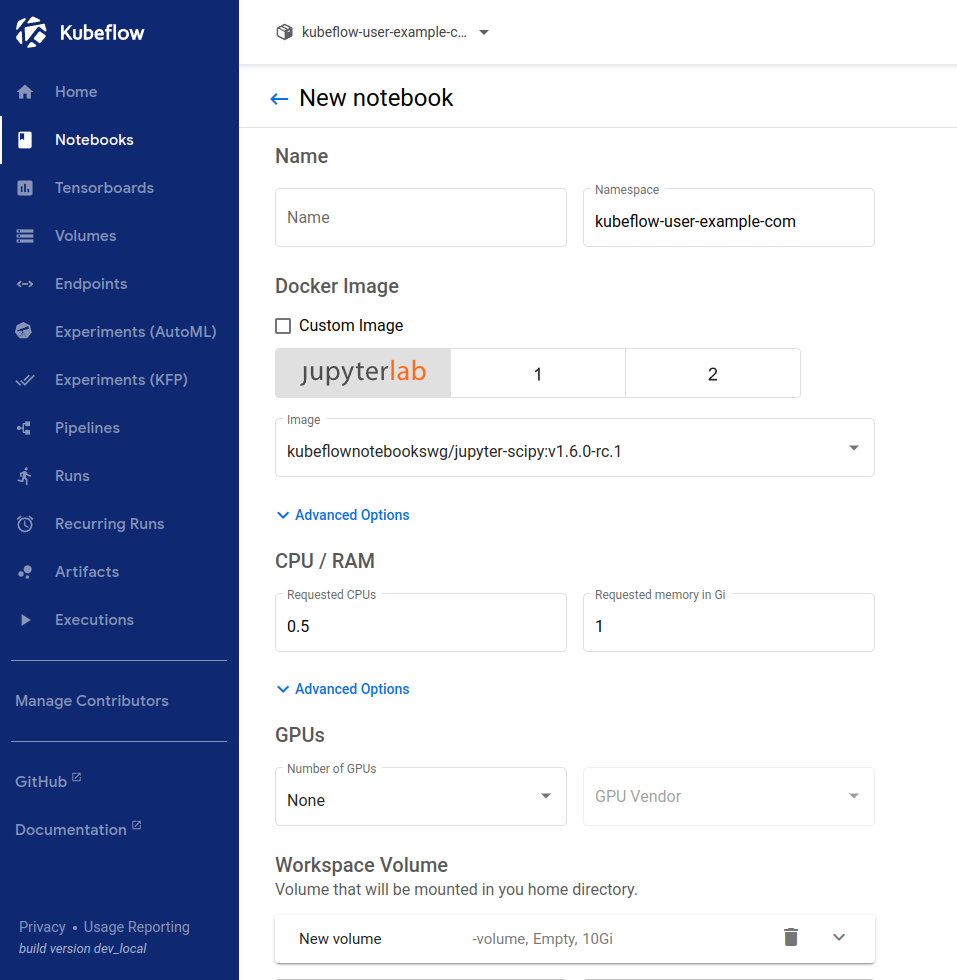
## بخش های مختلف kubeflow به همراه نحوه کار آنها

یکی از بخش های مورد توجه در kubeflow بخش notebooks می باشد. این بخش به شما این امکان را میدهد تا با روش های مختلف کدهای خود را طراحی و آزمایش کنید. در این بخش شما امکانات متفاوتی در اختیار دارید تا با آن بتوانید کدهای یادگیری ماشین خود را پیاده سازی کنید.

تصویر ۱



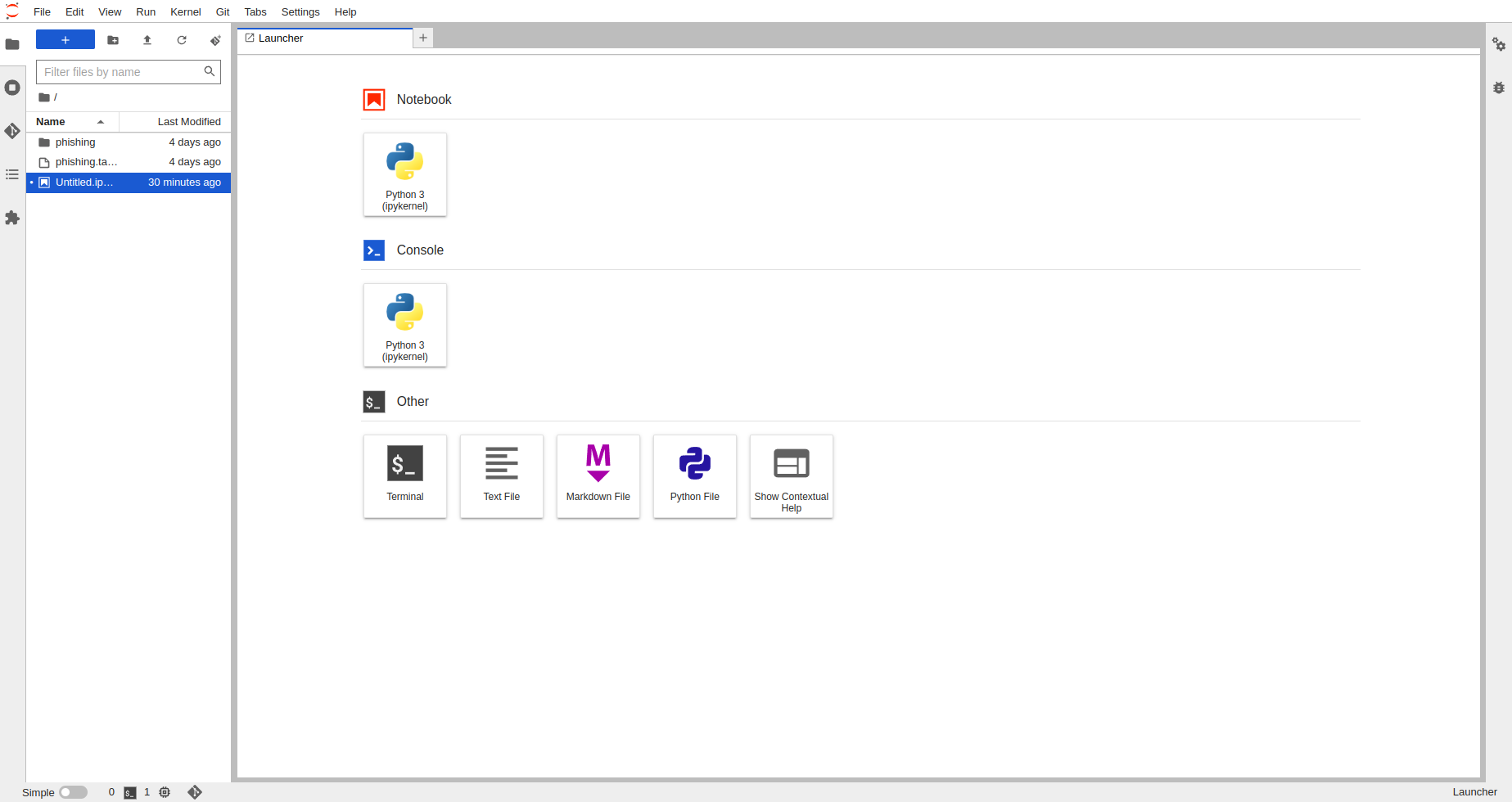
در این بخش شما میتوانید یک نوت بوک جهت آزمون و خطای کد های خود بسازید. این بخش به اینگونه عمل میکند که شما در ابتدا بر روی گزینه new notebook + کلیک میکنید و وارد بخش بعدی می شوید.



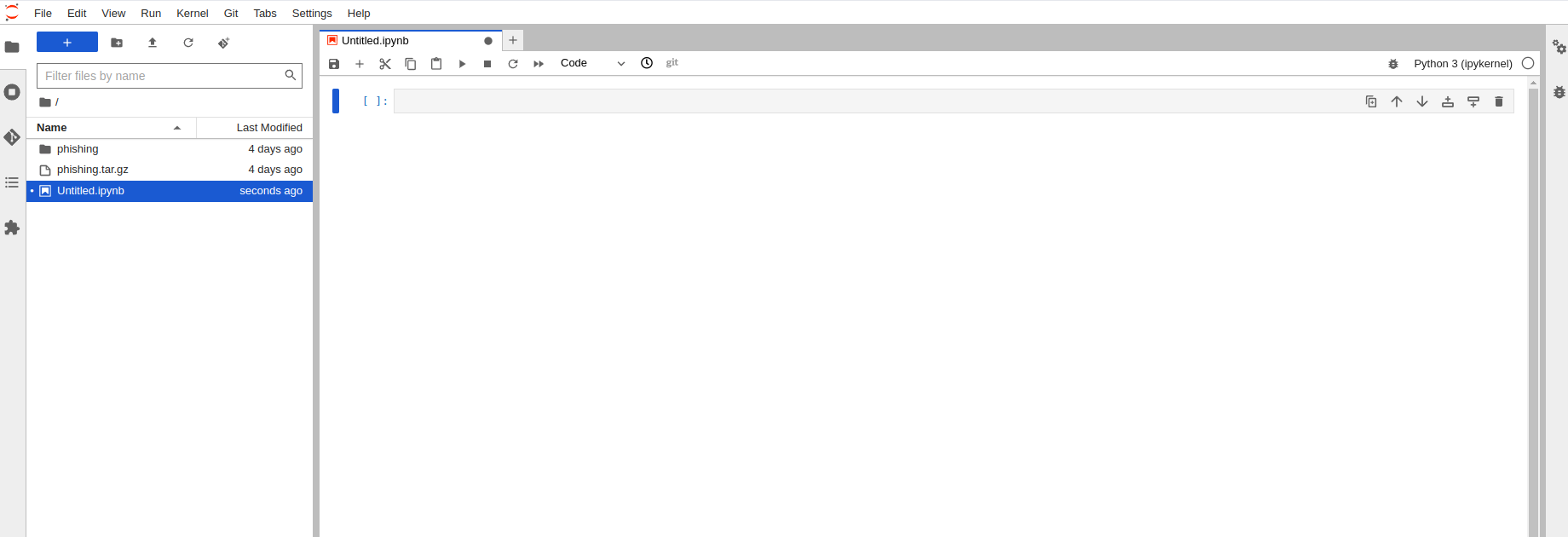
همانطور که در تصویر مشاهده میکنید شما در مرحله اول نام نوت بوک خود را انتخاب میکنید. در مرحله بعدی شما می توانید از docker image های آماده یا شخصی سازی شده استفاده کنید. Kubeflow تعدادی docker image آماده به صورت پیش فرض در اختیار ما قرار داده است. این مرحله در زمان اجرا و راه اندازی نوت بوک به طور پیش فرض انجام میشود. به طور مثال اگر شما بخواهید که کتابخانه tensorflow به همراه تعدادی از دیگر کتابخانه ها را در ابتدای این نوت بوک داشته باشید، می توانید در این بخش یا از image های آماده استفاده کنید یا به صورت شخصی سازی شده از image مورد نظر استفاده کنید.

همچنین می توانید تنظیمات پیشرفته تری برای image داشته باشید. در ادامه شما می توانید تنظیماتی از قبیل میزان فضای مصرفی یا دسترسی های لازم را جهت انجام اموری از جمله دسترسی به خط لوله به وسیله نوت بوک را انجام دهید. و در انتها شما می توانید تنظیمات متفرقه ای از جمله استفاده از حافظه اشتراکی را فعال یا غیر فعال کنید.

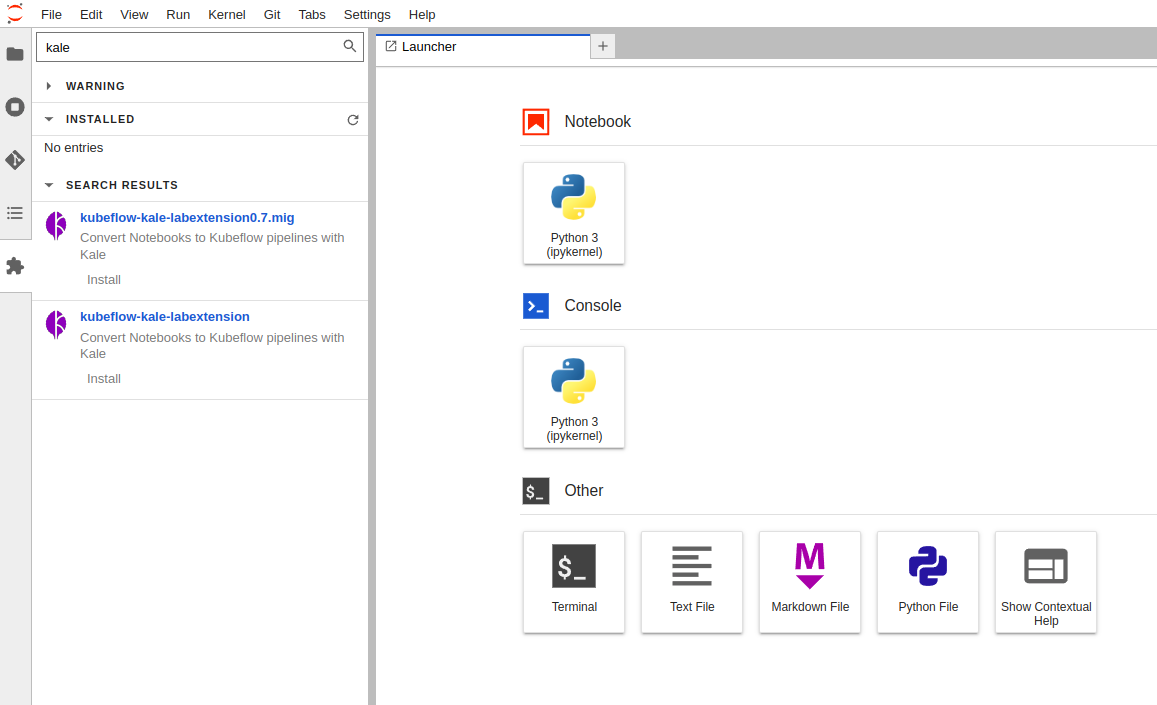
پس از ساخت یک نوت بوک جدید می توانید با کلیک بر روی connect وارد بخش بعدی شوید.



دراین بخش شما دسترسی های متفاوتی دارید. شما می توانید یک jupyter notebook ایجاد کنید و یا بر روی کنسول پایتون کدهای خود را اجرا کنید. از طرفی شما دسترسی های بسیاری دارید از جمله terminal را نیز دارید. این امر موجب میشود که شما بتوانید دستوراتی را در خط فرمان اجرا کنید. نحوه عملکرد این بخش به این صورت می باشد که شما یک نوت بوک جدید باز میکنید و می توانید کدهایی که میخواهید را دراین بخش آزمایش کنید.



همچنین شما می توانید کتابحانه های مورد نیاز خود نصب کنید و کدهای خود را اجرا کنید. شما در این بخش می توانید extention هایی را برای بهبود کار خود به نوت بوک اضافه کنید و این قابلیت برای این عمل در نوارابزار سمت چپ موجود می باشد. شما می توانید فایل هایی را در این بخش بارگذاری کنید یا فایل های ساخته شده خود را دانلود کنید.



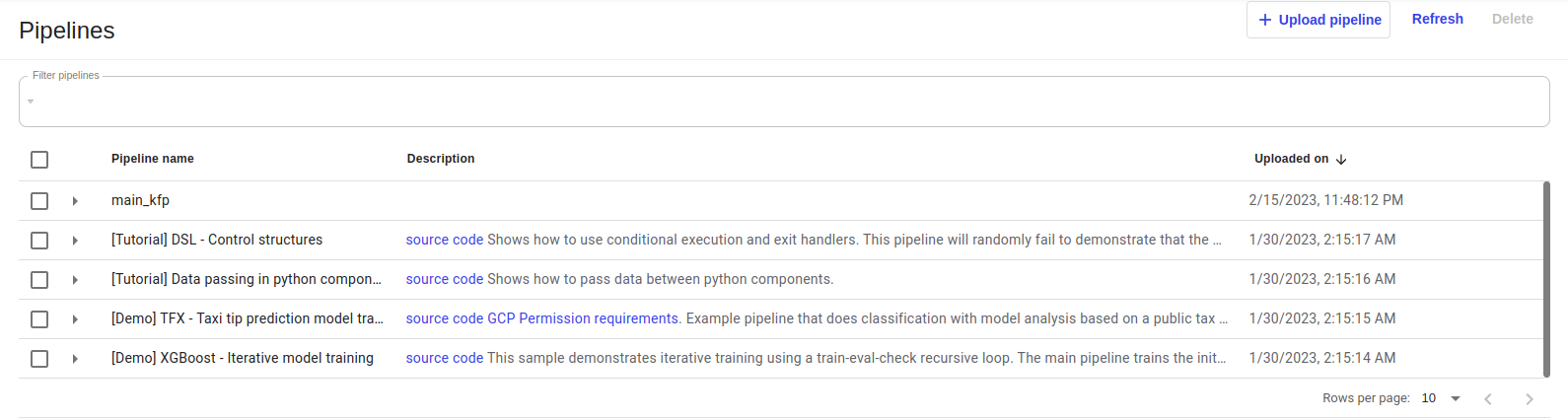
به علت متن باز بودن kubeflow برای این بخش extentions بسیاری طراحی شده است. به طور مثال kale که نوت بوک می باشد که قابلیت کامپایل کردن کدهای موجود در jupyter notebook را فراهم میکند.

از محدودیت های این بخش به این نکته میتوانیم اشاره کنیم که کدهای شما در یک فضای کاملا ایزوله طراحی می شود و امکان اتصال این کدها به خط لوله kubeflow به طور پیش فرض بر روی خود notebook موجود نمی باشد. به همین علت از extentions مثل kale استفاده میشود. Kale در ظاهر همانند نوت بوک معمولی می باشد اما قابلیت ایجاد خط لوله را فراهم میکند. به این صورت که شما می توانید هر قسمت از کدهای خود را به صورت مرحله ای از یک خط لوله به همراه وابستگی های مورد نظر معرفی کنید و در انتها آنها را در هسته اصلی kubeflow اجرا کنید. تمام مراحل به صورت گرافیکی می باشد.

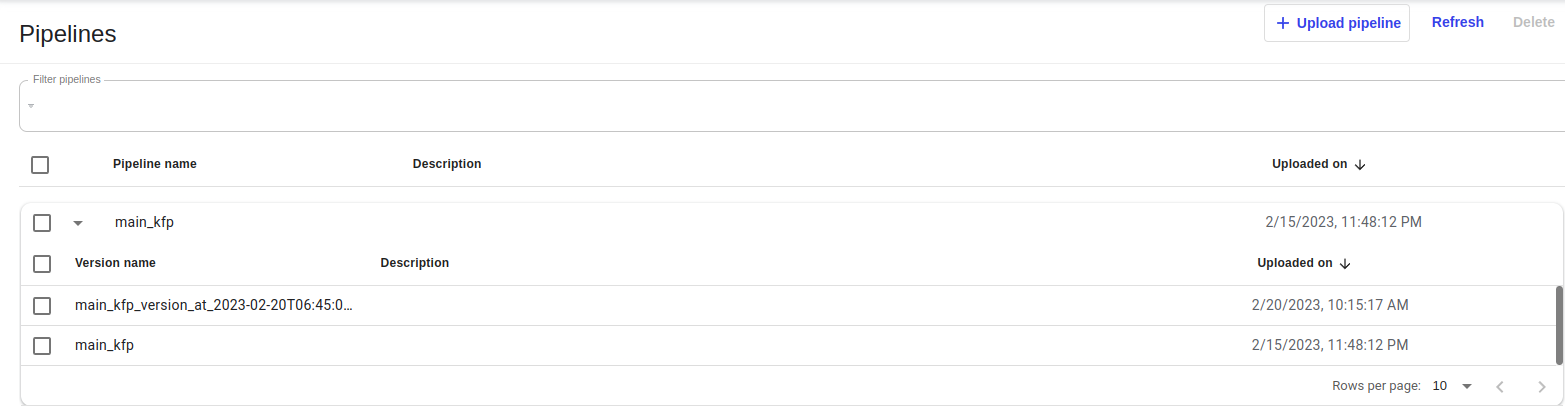
## خط لوله kubeflow

یکی از مهمترین بخش های kubeflow خط لوله می باشد. تمام فرایند ها برای اجرا شدن در kubeflow نیازمند یک خط لوله هستند. یعنی اگر یک گردش کاری بخواهد در kubeflow عمل کند نیازمند خط لوله می باشد.

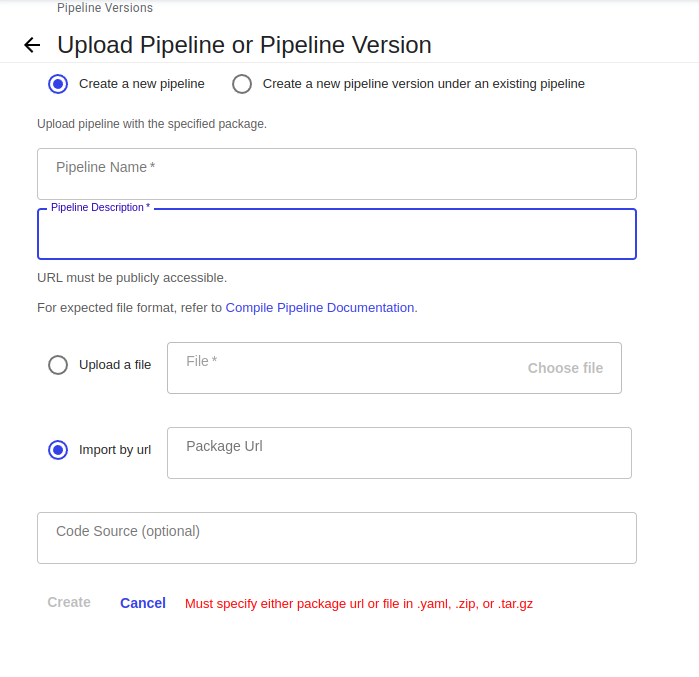
برای ساخت یک خط لوله ابتدا از نوار ابزار سمت چپ kubeflow بر روی pipeline کلیک میکنیم. تا وارد بخش pipeline شویم.



در kubeflow به طور پیش فرض تعدادی خط لوله برای نمونه های آماده وجود دارد که می توانیم از آنها استفاده کنیم. اما صرفا برای مشاهده نحوه عملکرد خط لوله می باشد. هر خط لوله می تواند دارای چندین اجرا باشد. یعنی هر خط لوله می تواند بارها اجرا شود و هر بار اجرا دارای ورژن می باشد.



ایجاد خط لوله: بر روی upload pipeline + کلیک میکنیم. وارد صفحه آپلود خط لوله میشویم.



همانطور که در تصویر مشاهده میکنید. این صفحه دارای بخش های مختلفی می باشد. که در ابتدای آن دو گزینه وجود دارد که نشان دهنده انتخاب خط لوله جدید و ورژن گذاری بر روی خط لوله موجود می باشد. اگر شما یک خط لوله داشته باشید که تنها تغییراتی بر روی کدهای موجود آن اعمال کرده اید نیاز دارید تا یک ورژن جدید بر روی کد زده شود تا اینکه یک خط لوله جدید ساخته شود. با توجه به نیاز شما گزینه مورد نظر در ابتدای کار انتخاب می شود.

اگر خط لوله جدید انتخاب شود ادامه فرایند نیازمند انتخاب یک نام برای خط لوله می باشد. پس از انتخاب یک نام شما باید یک فایل را بارگذاری یا از یک مسیر آن را انتخاب کنید. خط لوله ها فایل هایی با نوع yaml را اجرا میکنند.

نحوه ساخت فایل yaml:

یک خط لوله برای کار کردن نیازمند یک فایل yaml دارد که شامل کدها و همچنین نیازمندیها برای اجرا شدن هستند. در بخش notebook گفته شد که کدها کاملا مجزا از خط لوله طراحی میشوند و یک محیط کاملا تستی برای خود کدها می باشد. که در ادامه در خصوص kale گفته شد که یک محیط گرافیکی ایجاد میکند تا بتوان کدها را به صورت خط لوله اجرا کنیم. روش دیگر اجرای کدهای مورد نظر به وسیله SDK پایتون می باشد. در اصل افزونه kale نیز خود از کتابخانه های خاصی که kubeflow در اختیار ما قرار میدهد خط لوله را طراحی میکند تا کاربر با کدهای خاص برای اجرای خط لوله دست و پنجه نرم نکند.

از مزایای kale میتوانیم به این نکته اشاره کنیم که شما در خصوص SDK مورد نظر اطلاعات چندانی ندارید و تنها درگیر طراحی کدهای خود می باشید. و از معایب آن می توانیم به این نکته اشاره کنیم که کاملا فضا ایزوله و تستی می باشد و در ابعاد بزرگ قابل استفاده نیست. یا حتی به صورت یک پروداکت نمی تواند مورد استفاده قرار گیرد. صرفا محیط تستی می باشد.

مهم ترین بخش در طراحی خط لوله در kubeflow تبدیل کدهای خود به یک خط لوله می باشد، که در ادامه بخش های آن را توضیح میدهیم.

خط لوله kubeflow دارای دو ورژن می باشد (ورژن ۲ آن به صورت آزمایشی بوده و هنوز کاملا قابل استفاده نیست). که شامل فضای گرافیکی و کدنویسی است. فضای گرافیکی آن را در تصاویر قبل توضیح دادیم. اما خط لوله kubeflow دارای بخش خط لوله SDK نیز می باشد که آن را pipeline SDK می نامند.

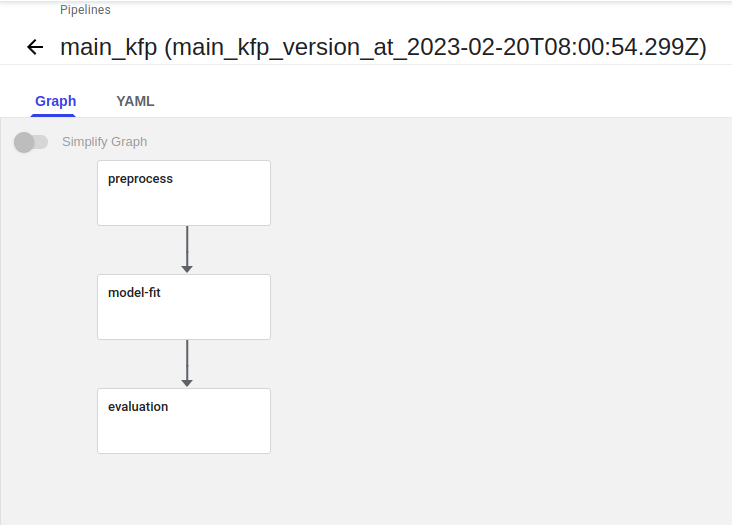
Kubeflow pipeline SDK

Kubeflow Pipelines SDK مجموعه‌ای از بسته‌های پایتون را ارائه می‌کند که می‌توانیم از آنها برای مشخص کردن و اجرای workflow یادگیری ماشین (ML) خود استفاده کنیم. خط لوله توصیفی از یک گردش کار ML است، شامل تمام اجزایی است که مراحل در گردش کار و نحوه تعامل اجزا با یکدیگر را تشکیل می دهند.

ما برای اینکه بتوانیم کدهای یادگیری ماشین خود را در قالب kubeflow pipeline sdk پیاده سازی کنیم نیازمند پکیج هایی هستیم که خود kubeflow برای ساخت خط لوله در اختیار ما گذاشته است. پکیج های مورد نظر در ادامه توضیح داده خواهد شد.

Kfp.compiler: کتابخانه kfp که برای خود kubeflow می باشد، که تبدیل کننده کدهای نوشته شده به فایل yaml است. در اصل پکیج kfp.compiler تبدیل کننده کدهای پایتون ما با کمک dsl به فایل های yaml است، و این فایل برای اجرای خط لوله لازم می باشد.

Kfp.component: هر تابع در کدهای ماشین لرنینگ ما به یک کامپوننت تبدیل میشود. در واقع خط لوله جریان کاری بین کامپوننت های ما می باشد. به طور مثال اگر مدل ما از ۳ تابع تشکیل شده است. تابع پیش پردازش، تابع یادگیری و تابع ارزیابی، در آن صورت ما یک خط لوله بین سه تابع خود نیاز داریم. یعنی ابتدا اطلاعات پیش پردازش شوند سپس اطلاعات برای مرحله یادگیری ارسال شوند و در نهایت مدل مورد ارزیابی قرار گیرد. این خط لوله به شکل زیر خواهد بود.



هر کدام از کامپوننت ها در اصل تبدیل به یک کانتینر شده و اجرا میشود. برای اینکه در کد بتوانیم این موضوع را تفکیک کنیم و خط لوله kubeflow این موضوع را متوجه شود از این پکیج استفاده میکنیم. تمام تعامل ها در خط لوله بین توابع و کلاس ها با این پکیج صورت میگیرد. این پکیج دارای زیر مجموعه هایی می باشد که به وسیله آنها می توانید کامپوننت ها را به روش های مختلفی تعریف کنید.

Kfp.dsl: حاوی زبان مخصوص دامنه (DSL) است که می توانیم از آن برای تعریف تعامل با خطوط لوله و مولفه ها استفاده کنیم.

Kfp.client: این پکیج شامل کتابخانه های kubeflow pipeline API می باشد. در مراحل قبلی وقتی ما یک کد ماشین لرنینگ را کامپایل میکنیم در نهایت یک فایل yaml برای ما ایجاد میکند و آن فایل را در بخش ایجاد خط لوله آپلود میکنیم تا خط لوله ایجاد شود. اما اگر در فرایند پورداکت باشیم و بخواهیم که خط لوله بدون بارگذاری فایل yaml به صورت دستی صورت گیرد از این پکیج استفاده میکنیم. این پکیج این امکان را میدهد تا ما از داخل فضای کد مستقیما خط لوله را ایجاد و آن را اجرا کنیم.

اگر مراحل را به طور کلی بنگریم به این صورت می باشد که ابتدا ما کدهای خود را به صورت توابعی طراحی میکنیم که می توانند کاملا از یکدیگر مجزا باشند. سپس هر تابع را به صورت یک کامپوننت تعریف میکنیم و این کامپوننت ها را به وسیله یک تابع که آن را خط لوله می نامیم به یکدیگر متصل میکنیم. متدهای تبدیل کننده یک تابع به خط لوله در پکیج kfp.dsl می باشد. این پکیج در اصل متصل کننده و ایجاد کننده خط لوله پروژه می باشد. در انتها این خط لوله یا به وسیله کامپایلر اجرا شده و یک خروجی فایل yaml از آن ساخته می شود یا در حالت بعدی به وسیله kfp.client به صورت مستقیم اجرا می شود.

نکته: هر کامپوننت برای اجرا شدن درون خود یک فایل yaml می سازد تا بتواند آنها را اجرا کند. برای این منظور که کامپوننت ها اجرا شوند دو روش وجود دارد

* استفاده از image هایی که از قبل گرفته شده است.
* بارگذاری کدهای درون فایل yaml برای اجرا.

در روش اول کامپوننت ها به صورت کاملا مجزا اجرا شده و image هر کدام را میگیریم. در این مرحله می توانیم آنها را در repository شخصی یا عمومی قرار دهیم و آدرس آن را در تابع خط لوله فراخوانی کنیم. که این روش در زمان اجرا ابتدا image کامپوننت را فراخوانی کرده و سپس آن را اجرا میکند.

در روش دوم به وسیله کامپایلر کدهای کامپوننت ها را درون خود فایل yaml ذخیره میکنیم تا در هنگام اجرا نیاز به دریافت image از repository خاصی نباشد. هر دو روش قابل اجرا و تست می باشد.

مشکلاتی که در روش اول با آنها روبرو میشویم این می باشد که امکان دارد repository ها در هنگام دسترسی با kubeflow دچار مشکل شود. و از طرفی دیگر توسعه این مدل ها دارای روند بسیار طولانی تری می باشد زیرا ابتدا باید image گرفته و در repository ذخیره شود. در نهایت آدرس در فایل yaml قرار گیرد. این موضوع توسعه را تا حدی بسیاری کند میکند.

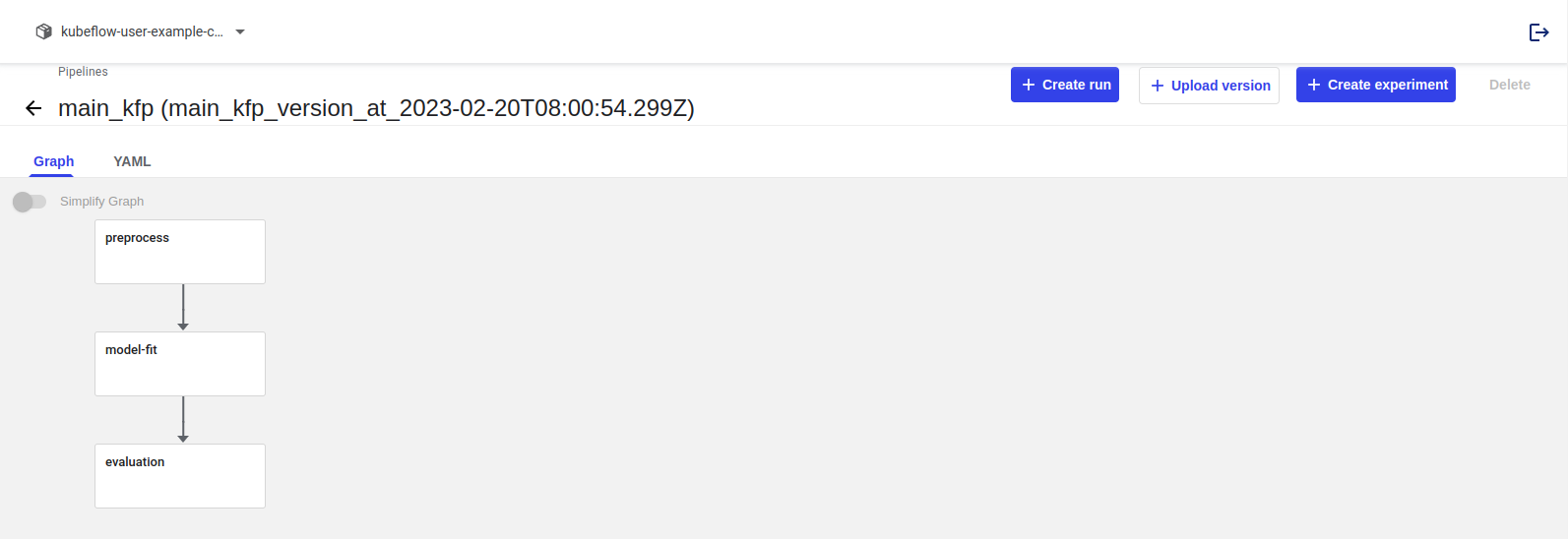
اما از مزایای این روش می توان به این نکته توجه نمود که می توانیم repository شخصی خود را معرفی کرده و از تمام کامپوننت ها یک image داشته باشیم که پایداری کامپوننت ها را تا حدود بسیار زیادی حفظ میکند و وابستگی ها را کاهش میدهد.

مزایایی که در روش دوم وجود دارد، توسعه را بسیار سریع می کند و نیاز نیست که حتما image ها در مکان خاصی ذخیره شده باشند و کدها همراه با فایل yaml می باشد و در هنگام ایجاد خط لوله در لحظه image را می سازد. این امر سرعت فاز توسعه را افزایش می دهد.

در تصویر خط لوله بالا یک نمونه از خط لوله اجرا شده به وسیله روش دوم که کدها درون yaml قرار دارند ساخته شده است.

## ایجاد و بارگذاری خط لوله

در بخش قبلی هنگامی که ما خط لوله خود را ایجاد و بارگذاری کردیم به مرحله اجرا می رویم.

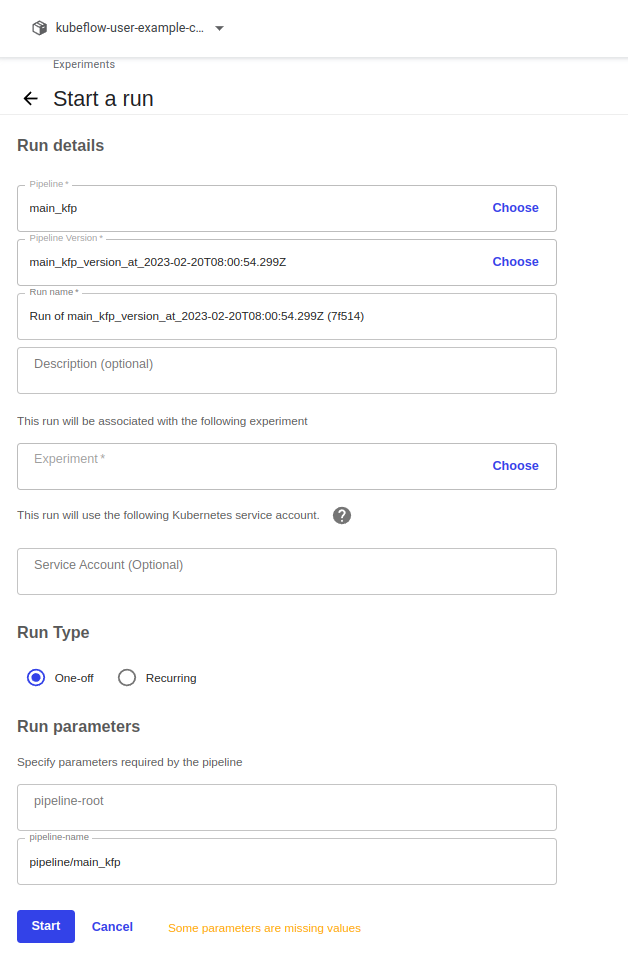


برای اجرای یک خط لوله بر روی گزینه create run + در سمت راست و بالای تصویر کلیک میکنیم تا به مرحله ساخت یک اجرا برویم. در اصل هر خط لوله باید یک مرحله اجرا داشته باشد و بدون اجرا ما تنها خط لوله ایجاد کرده ایم.

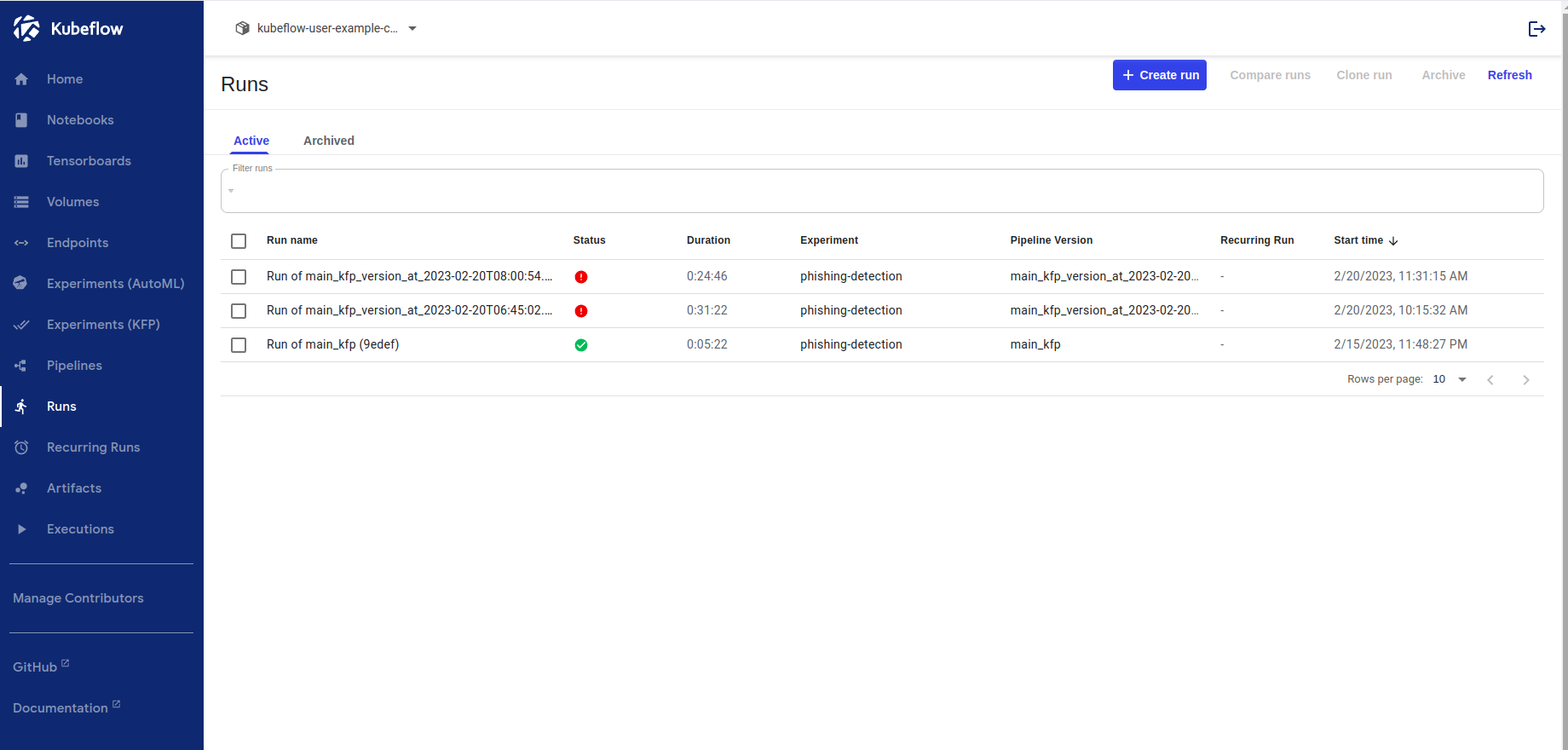
نکته:‌برای اجرای خط لوله دو روش وجود دارد ابتدا خط لوله را با فایل yaml آپلود کنیم و به مرحله ی فعلی وارد شده و به صورت کاملا گرافیکی خط لوله را اجرا کنیم.

در روش دوم، که در مرحله خط لوله آن را توضیح دادیم، می توانیم درون کد با پکیج kfp.client به صورت کاملا خودکار این کار را انجام دهیم.

در این مرحله ما اجرا یک خط لوله را کاملا به صورت گرافیکی توضیح میدهیم.

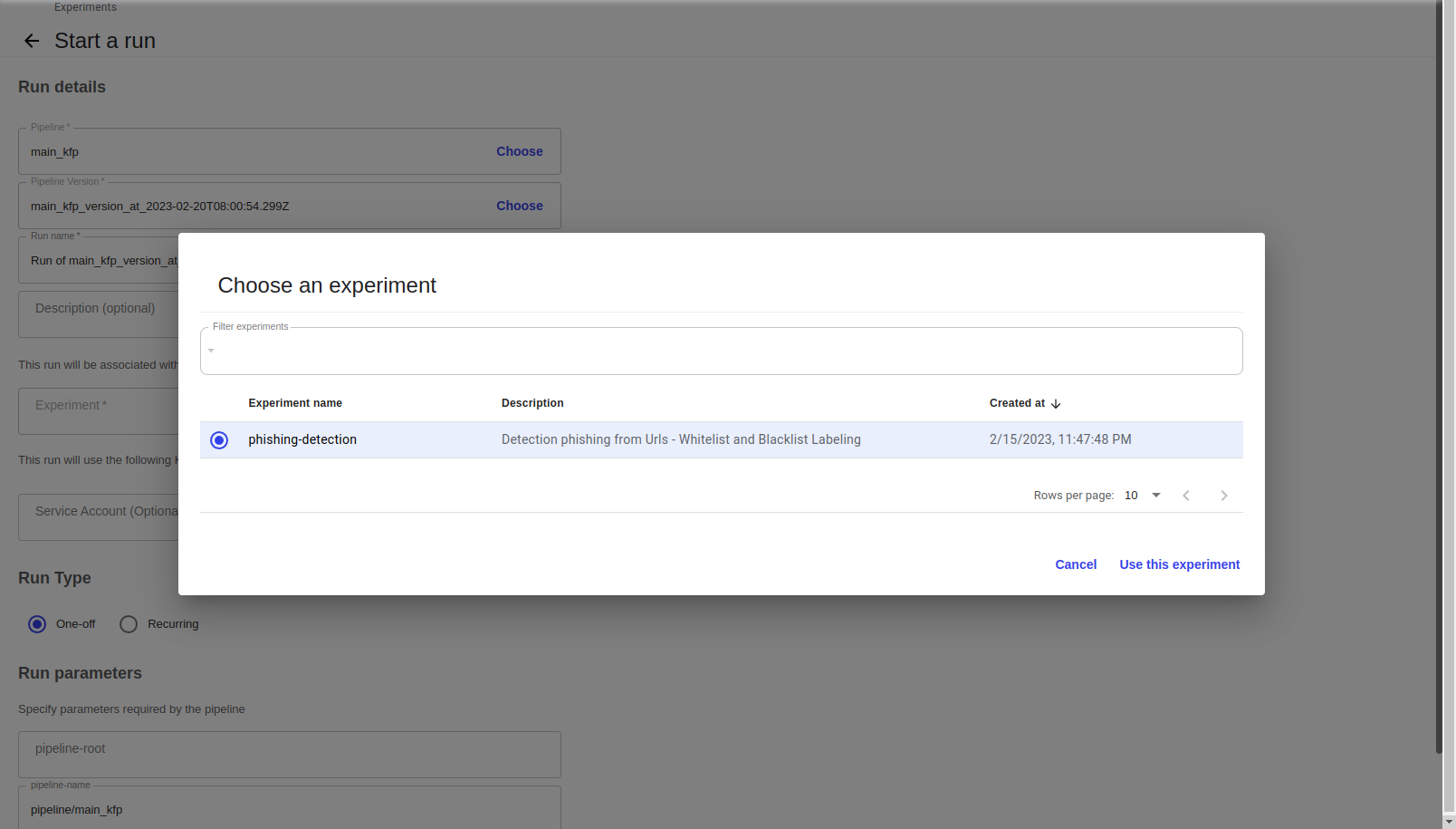


پس از کلیک بر روی create run + شما وارد مرحله بعدی می شوید، که دارای بخش های مختلفی می باشد. این بخش خط لوله مورد نظر را انتخاب کرده و همچنین ورژن آن را نیز آماده دارد، البته اگر از منوی سمت چپ صفحه اصلی kubeflow بر روی Runs کلیک کنید، لیست تمام اجرا هایی که تا آن زمان صورت گرفته را مشاهده میکنید.



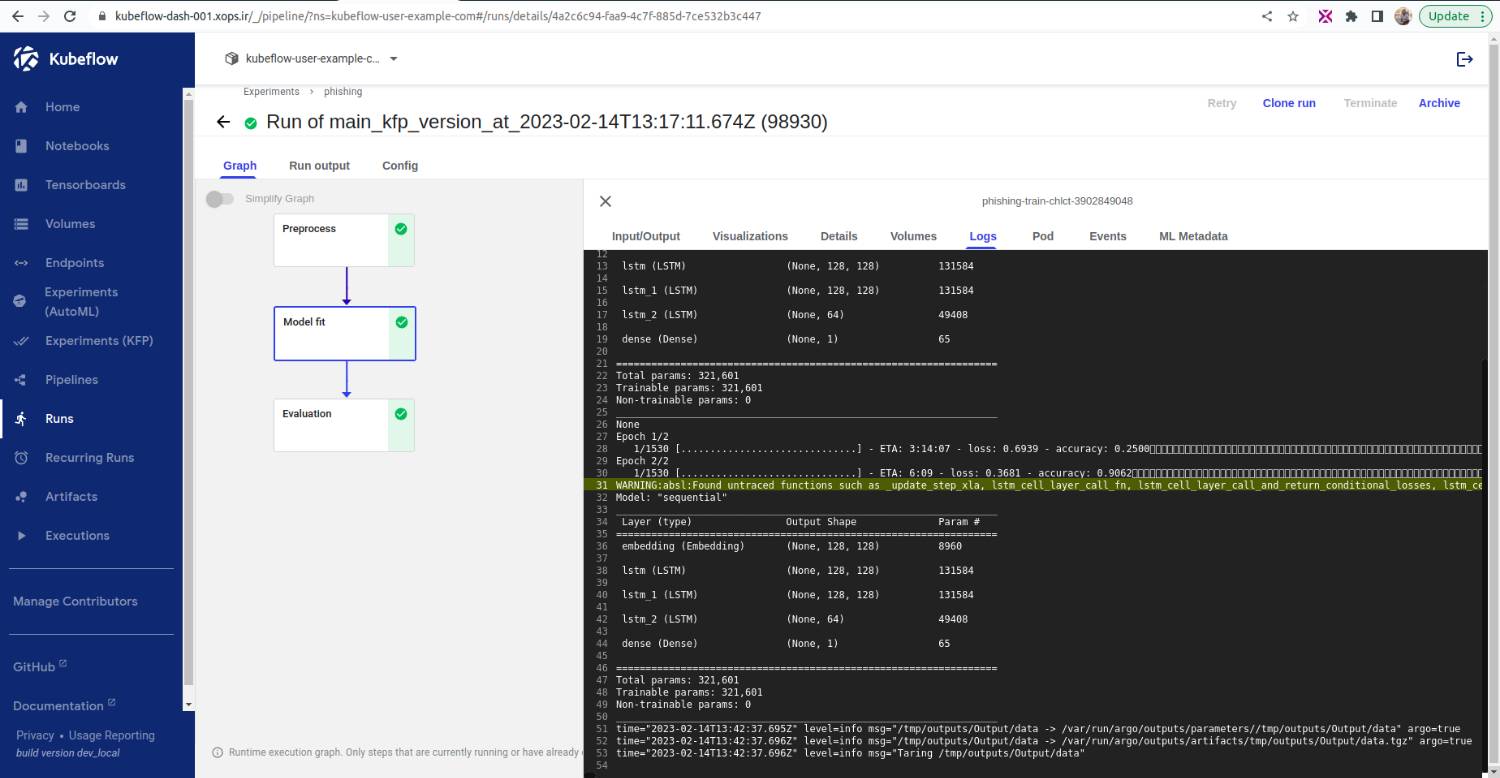
همانطور که مشاهده میکنید تعداد اجراها، حالت ها، زمان، فضای تجربه، ورژن و زمان آنها را می توانید مشاهده کنید. اطلاعات تمام اجراهای ممکن دراین بخش موجود بوده و به صورت آرشیو ذخیره می شود. اگر از این بخش هم بر روی create run + کلیک کنیم وارد بخش اجراها که در تصویر قبل در خصوص آن صحبت کردیم می شویم.

در صفحه ساخت یک اجرا نام های مورد نظر را وارد میکنید اما اگر از بخش pipelines به صفحه ساخت اجرا رفته اید نیاز به وارد کردن اطلاعات نیست و می توانید آنها را تنها ویرایش کنید. هر اجرا نیاز به یک experiment دارد. که شما می توانید یک تجربه در بخش تجربه ها ایجاد کنید و در این بخش آن را انتخاب کنید. نکته قابل ذکر این می باشد که برای هر بار اجرا یا ورژن گذاری نیاز به انتخاب چندین تجربه نمی باشد. تنها با یکبار ایجاد experiment شما می توانید تمام اجراهای خود را در آن قرار دهید. صفحه تجربه در بخش بعد توضیح داده خواهد شد. نحوه انتخاب experiment در اجراها به صورت زیر می باشد.



پس از تکمیل کردن اطلاعات شما می توانید نوع اجرا را مشخص کنید که تنها یک بار اجرا شود یا چندین بار به صورت دوره ای اجرا شود. و در انتها پارامترهای مورد نیاز برای اجرای خط لوله را می توانید در بخش نهایید وارد کنید.

یک نمونه از اجرای موفقیت آمیز مدل هوض مصنوعی phishing در خط لوله kubeflow به صورت زیر می باشد.

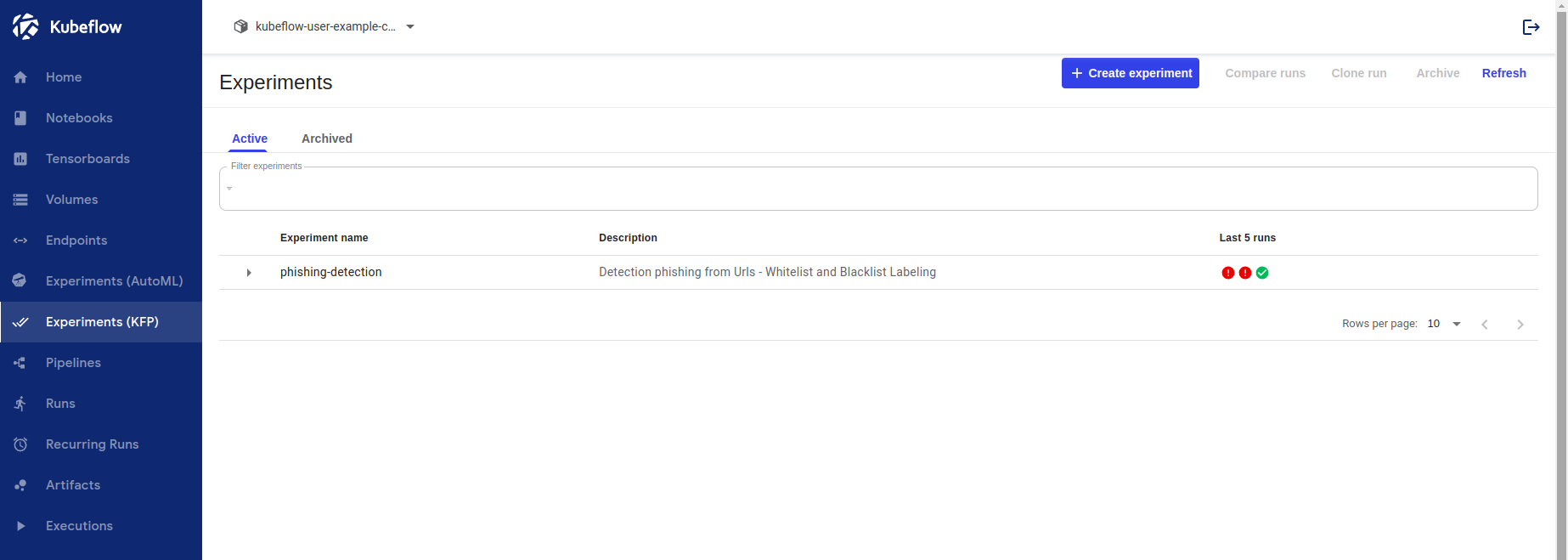


قسمت های مختلفی در بخش اجرا وجود دارد که تمام آنها را می توانیم با کمک پکیج ها درون کد یادگیری ماشین خود قرار دهیم و خروجی ها را مشاهده کنیم. به طور مثال می توانیم روند آموزش، منحنی ROC یا دیگر متریک های اندازه گیری را در هنگام یادگیری مدل مورد ارزیابی قرار دهیم.

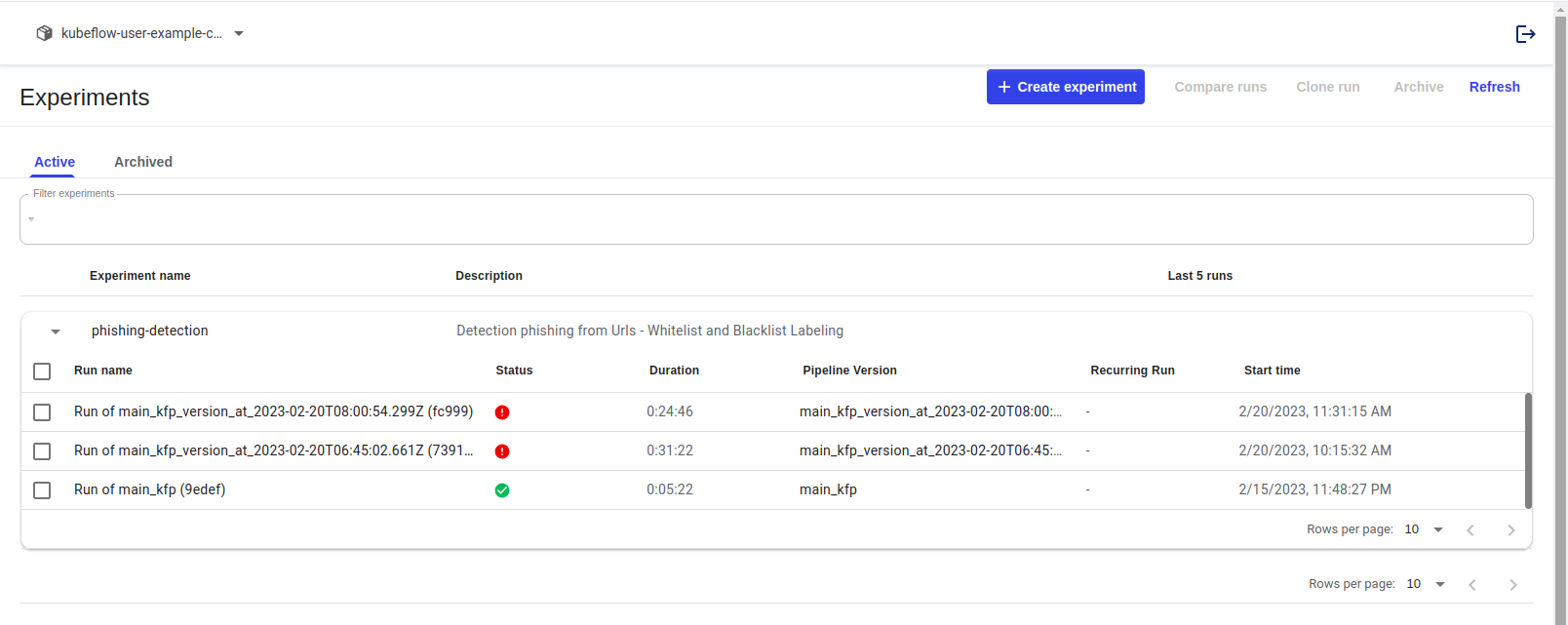
نکته: هنگامی که یک خط لوله به اتمام می رسد ورژن گذاری شده و اطلاعات آن درج می شود اما برای اینکه بتوانیم از مدلی که در این خط لوله آموزش دیده استفاده کنیم. روش های مختلفی وجود دارد که یکی از آن ذخیره سازی مدل می باشد. ما می توانیم بعد از اینکه آموزش دید آن مدل را ذخیره سازی کنیم. البته نوع ذخیره سازی کاملا اختیاری می باشد. برای phishing مدل آموزش دیده بر روی s3 ذخیره شد.

## تجربه اجرای یک خط لوله (Experiments kfp)

در این بخش تجربه هایی که به صورت اجرای یک خط لوله صورت گرفته است را به صورت مجتمع مشاهده میکنید. همچنین برای اجرای هر خط لوله شما نیازمند یک experiment می باشید. اما می توانیم برای یک پروژه مثل phishing از یک experiment برای چندین خط لوله و چندین اجرا برای هر خط لوله استفاده کنیم. صفحه experiments kfp به این صورت می باشد.



که می توانیم جزئیات هر اجرا را نیز مشاهده کنیم.



## Endpoint

در مرحله نهایی فرض می شود که خط لوله در حال اجرا می باشد و به صورت دوره ای در حال یادگیری است، و مدل همواره در حال ذخیره سازی و بروز رسانی می باشد.

Endpoint این اجازه را به ما می دهد تا بتوانیم از بیرون کیفیت مدل ها را ارزیابی کرده یا از مدل هایی که طراحی می شوند خروجی تهیه کنیم.

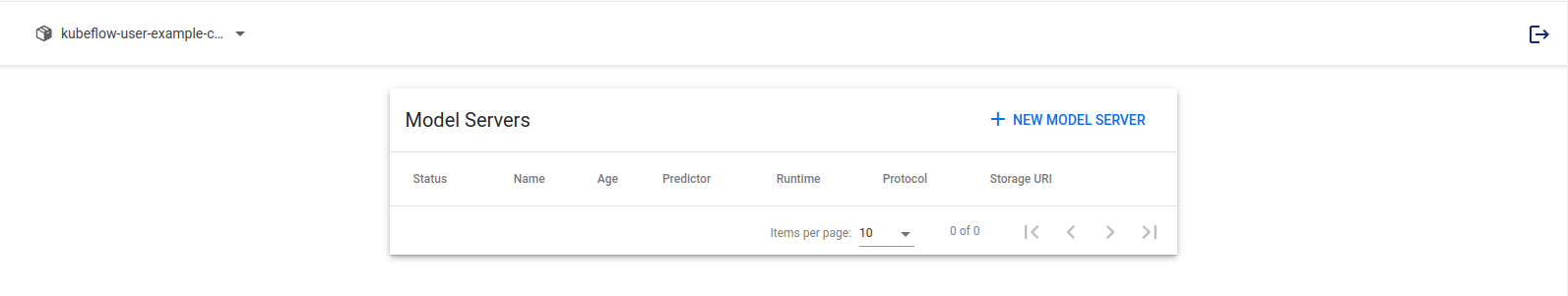
یکی از روش هایی که به وسیله آن می توانیم از Endpoint در kubeflow استفاده کنیم، kserve می باشد، که به صورت یک افزونه بیرونی است. و به وسیله خود kubeflow طراحی شده است.

## Kserve

یک webapp برای مدیریت سرورها در Endpoint است. به وسیله رابط kserve یک کاربر می تواند مدل هایی که در kubeflow طراحی و ذخیره سازی شده اند را دستکاری میکند. Kserve به صورت یک افزونه می تواند بر روی kubeflow نصب شود و پس از آن در بخش Endpoint قابل دسترسی باشد.

این بخش نیز در خود kubeflow یک فرایند کاملا گرافیکی دارد و همچنین می توانیم با کمک افزونه ها که یک نمونه از آن را معرفی کردیم طراحی کنیم.

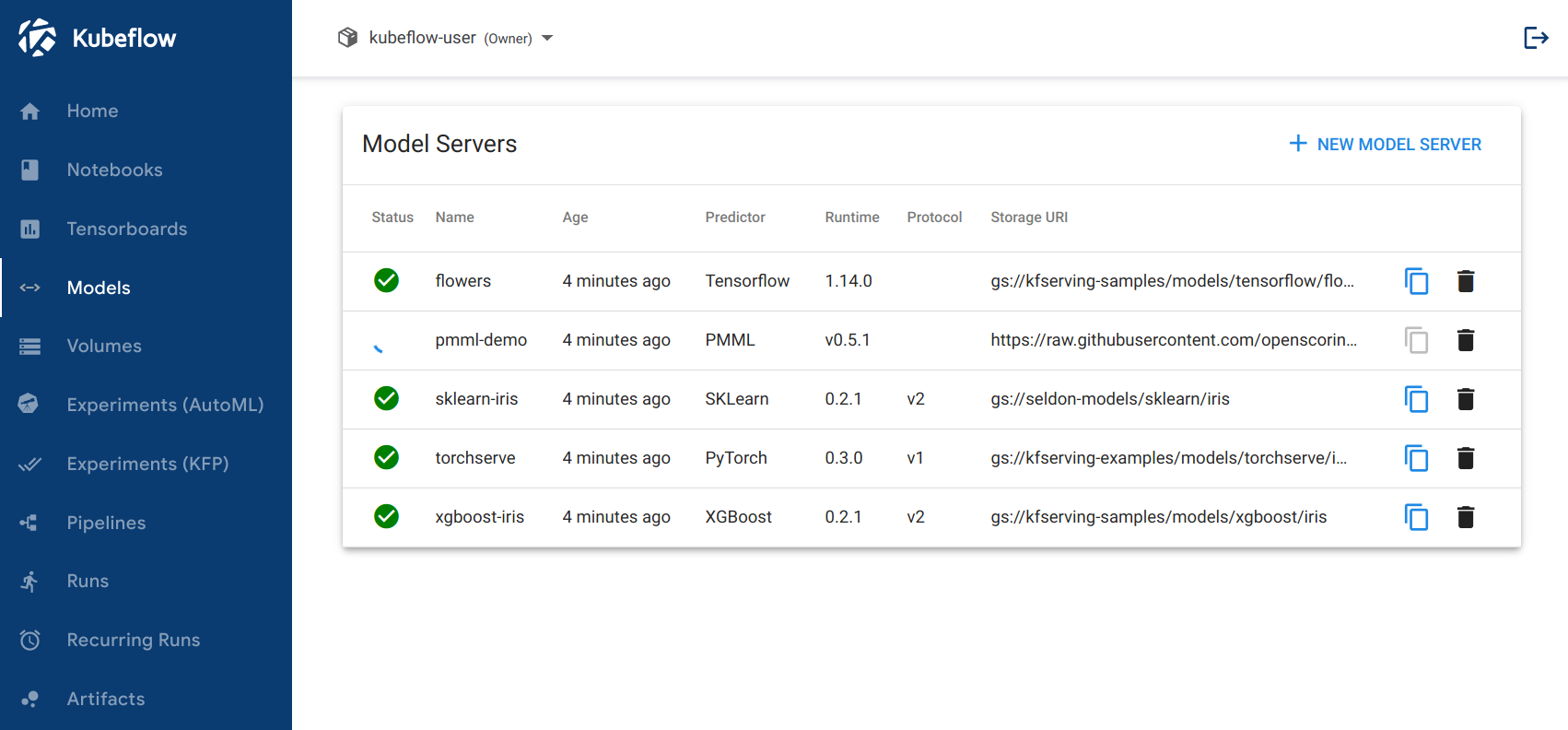
اگر بر روی endpoint در صفحه اصلی kubeflow کلیک کنید تصویر زیر را مشاهده خواهید کرد.



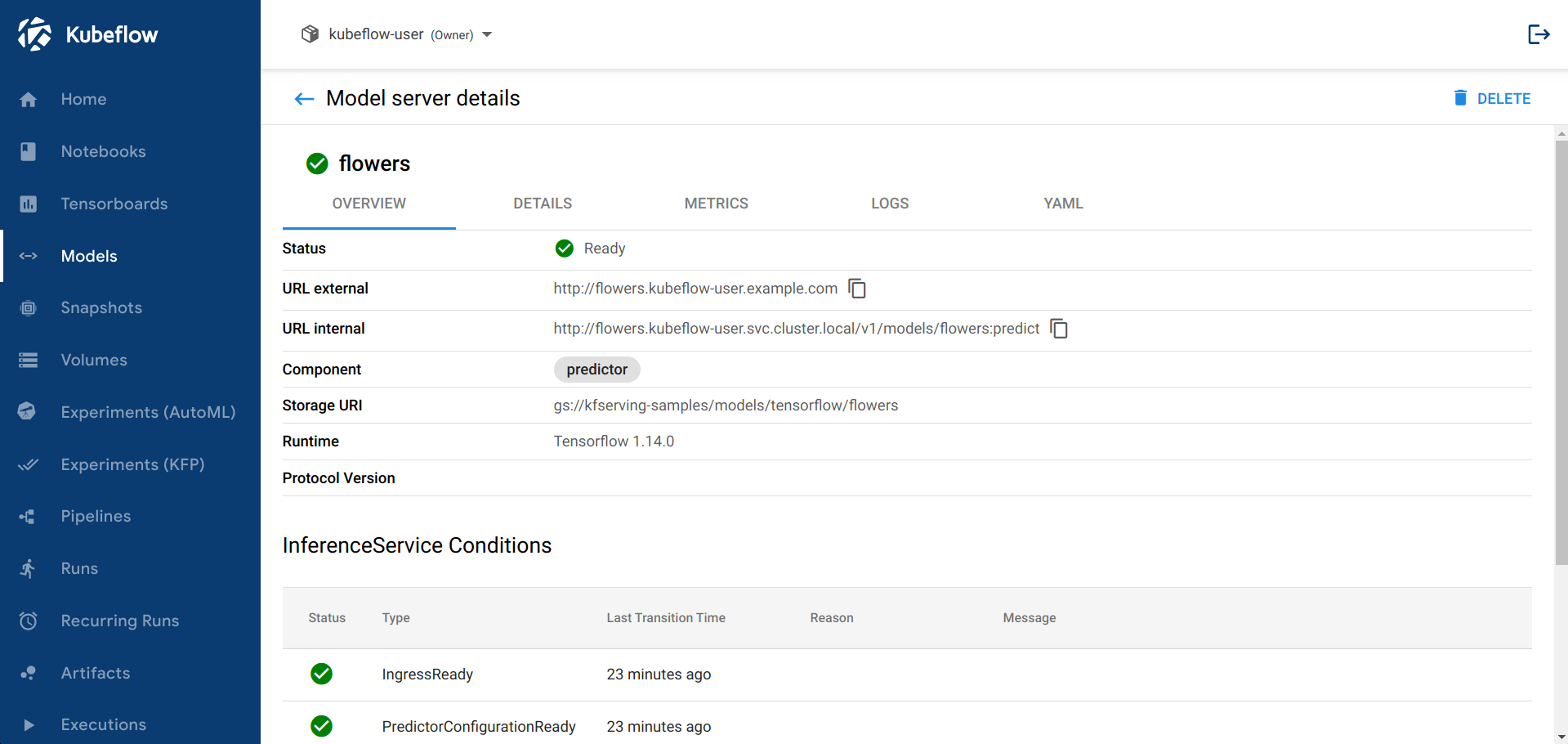
برای ایجاد یک مدل سرور بر روی new model server + کلیک کرده و به صفحه بعد می روید. که نیازمند وارد کردن یک تنظیمات yaml برای سرو کردن مدل می باشد.

برای سرو کردن مدل شما می توانید از فایل های yaml آماده که مدل هایی را که به صورت استاندارد طراحی شده و سرور آنها نیز از قبل موجود است استفاده کنید تا نحوه عملکرد این بخش را آزمایش کنید.

در این تصویر تعدادی از سرورهایی که به صورت آماده خود kubeflow به نمایش گذاشته است را مشاهده میکنید.



شما می توانید به راحتی این سرورها را حذف کنید. یا از نو بسازید. همچنین می توانید جزئیاتی از آنها را نیز مشاهده کنید.



اما اگر ما بخواهیم از مدل سرور های اختصاصی استفاده کنیم می توانیم از قابلیت های kserve برای این منظور استفاده کنیم.

KServe یک Kubernetes CRD ساده را برای فعال کردن استقرار مدل‌های آموزش‌دیده منفرد یا چندگانه در زمان‌های اجرای سرویس ها فراهم می‌کند. علاوه بر این MLServer پروتکل پیش‌بینی v2 را با REST و gRPC پیاده‌سازی می‌کند. آنها می‌توانند سرویس‌دهی مدل خارج از فضای kubeflow را ارائه دهند، اما ما همچنین می‌توانیم سرور مدل خود را برای موارد پیچیده‌تر بسازیم. KServe مقدمات اولیه API را فراهم می کند تا به ما امکان دهد زمان اجرای سرویس مدل سفارشی را به راحتی بسازیم، همچنین می توانیم از ابزارهای دیگری مانند BentoML برای ساخت تصویر سرویس مدل سفارشی خود استفاده کنیم.

برای ساخت یک سرور اختصاصی در kserve نیاز می باشد که ابتدا یک سرور با استفاده از پکیج kserve طراحی کنیم. سپس این سرور می تواند از طریق REST و یا gRPC با بیرون ارتباط برقرار کند. پکیج kserve به صورت خودکار این فرایند API را هندل میکند. و ما تنها نیاز می باشد که یک سرور را طراحی کنیم و با استفاده از متدهای معرفی شده در پکیج ورودی و خروجی های لازم را تنظیم کنیم. سپس این کد را می توانیم به صورت یک فایل image ذخیره کرده و یک فایل yaml را برای Endpoint در kubeflow طراحی کنیم.

یک نمونه از سرورهایی که برای مدل phishing طراحی کردیم به این صورت می باشد:

from typing import Dict

from kserve import Model, ModelServer

def pre\_process(sample: list):

from keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

max\_chars = 20000

max\_len = 128

tokenizer = Tokenizer(num\_words=max\_chars, char\_level=True)

tokenizer.fit\_on\_texts(sample)

sequences = tokenizer.texts\_to\_sequences(sample)

data = pad\_sequences(sequences, maxlen=max\_len)

return data

class Phishing(Model):

def \_\_init\_\_(self, name: str):

super().\_\_init\_\_(name)

self.model\_ = None

self.name = name

self.load()

def load(self):

import s3fs

import zipfile

import tempfile

import keras

with tempfile.TemporaryDirectory() as tempdir:

s3fs = s3fs.S3FileSystem(

key="minio",

secret="minio123",

client\_kwargs={

'endpoint\_url': "https://minio-kubeflow.xops.ir/"

}

)

s3fs.get(f"minio/data/phishing/model.zip", f"{tempdir}/model.zip")

with zipfile.ZipFile(f"{tempdir}/model.zip") as zip\_ref:

zip\_ref.extractall(f"{tempdir}/model")

model\_ = keras.models.load\_model(f"{tempdir}/model")

self.model\_ = model\_

def predict(self, payload: Dict, headers: Dict[str, str] = None) -> Dict:

url\_str = payload["instances"][0]["url"]

data = pre\_process(sample=[url\_str])

result = self.model\_.predict([data])

return {"predictions": result[0, 0]}

if \_\_name\_\_ == "\_\_main\_\_":

model = Phishing("custom-model")

ModelServer().start([model])

در این کد همانطور که مشاهده میکنید یک کلاس با نام phishing طراحی شده است، که از کلاس kserve.model ارث بری کرده است. برای طراحی یک سرور نیازمند این ارث بری می باشیم. در مرحله اول کاملا ساده مدل را فراخوانی کرده ایم. این مدل در یک s3 ذخیره شده است و ما آن را دریافت و آماده سرو کرده ایم. در مرحله بعد در متدی با عنوان predict که یکی از نیازمندی های kserve می باشد دیتاهای بیرونی را دریافت میکنیم. با کمی پیش پردازش بر روی داده ها آنها را برای پیش بینی به مدل ذخیره شده خودمان میدهیم و خروجی نهایی را به صورت جیسان بر میگردانیم. این یک نمونه ساده از مدل سرور ما می باشد.

برای اجرای این مدل سرور می توانیم آن را به صورت image در یک repository ذخیره کنیم. در مرحله بعدی یک فایل yaml به صورت زیر طراحی کنیم:

apiVersion: serving.kserve.io/v1beta1

kind: InferenceService

metadata:

name: custom-model

spec:

predictor:

containers:

- name: kserve-container

image: ${DOCKER\_USER}/custom-model:v1

در این فایل yaml ما در جای image آدرس repository خود به همراه image مورد نظر را جایگزین میکنیم. سپس می توانیم این فایل yaml را در kubeflow در قسمت Endpoint جایگذاری کنیم تا مدل سرور اختصاصی ما آماده سرو کردن شوند.

به این ترتیب ما می توانیم مدل سرور خود را کاملا شخصی سازی شده استفاده کنیم.

پکیج kserve متن باز بوده و می توانیم به هر شکلی که مایل هستیم از آن استفاده کنیم. از طرفی می توانیم به جای استفاده از kserve خود از REST یا gRPC برای رابط api استفاده کنیم. اما با توجه به جامع بودن مدل kserve بهترین روش استفاده از این مدل می باشد. اما همانطور که مشاهده میکنید این مدل کاملا انعطاف پذیر بوده و امکان هر گونه تغییر را برای سرور ها فراهم میکند.

# منابع موردنیاز

# زمان‌بندی و شکست کارها

پیاده‌سازی MLOps معمولاً شامل مراحل زیر است:

* توسعه مدل: با استفاده از ابزارهایی مانند Jupyter Notebook، TensorFlow، PyTorch و غیره، مدل‌های یادگیری ماشین را توسعه و آموزش دهید.
* کنترل نسخه مدل: نسخه‌های مدل را با استفاده از سیستم‌های کنترل نسخه مانند Git ذخیره و مدیریت کنید.
* تست مدل: مدل‌ها را با استفاده از تست‌های خودکار آزمایش کنید تا مطمئن شوید که الزامات عملکرد و دقت را برآورده می‌کنند.
* استقرار مدل: مدل‌ها را با استفاده از زیرساخت‌ها به عنوان ابزارهای کد مانند Terraform یا CloudFormation در تولید مستقر کنید.
* مانیتورینگ مدل: مدل‌های در حال تولید را با استفاده از ابزارهایی مانند TensorFlow Serving، Seldon یا KFServing جهت شناسایی و پاسخگویی به مسائل نظارت کنید.
* تعمیر و نگهداری مدل: به طور مرتب مدل‌ها را به روز رسانی و نگهداری کنید تا عملکرد آنها را بهبود بخشید و مشکلاتی را که پیش می‌آید برطرف کنید.
* همکاری: تشویق به همکاری بین دانشمندان داده، توسعه دهندگان و تیم‌های عملیاتی جهت اطمینان از MLOهای روان و مؤثر.
* یکپارچه سازی مداوم و استقرار مستمر (CI/CD): فرآیند یکپارچه سازی، آزمایش و استقرار مدل‌ها را با استفاده از ابزارهایی مانند جنکینز یا GitLab به صورت خودکار انجام دهید.
* قابلیت توضیح مدل: اطمینان حاصل شود که مدل‌ها شفاف و قابل توضیح هستند، به طوری که تصمیمات آنها قابل درک و اعتماد توسط ذینفعان باشد.

مراحل اجرای Kubeflow معمولاً شامل موارد زیر است:

* آماده سازی: جمع‌آوری نیازمندی‌ها، ارزیابی زیرساخت‌ها و تعیین محدوده پروژه اجرای Kubeflow.
* نصب: Kubeflow را روی زیرساخت هدف، از جمله خوشه‌های Kubernetes و هر گونه وابستگی لازم، نصب و پیکربندی کنید.
* آماده‌سازی داده‌ها: داده‌ها را جهت استفاده در Kubeflow آماده و سازماندهی کنید، از جمله داده‌ها، پاک‌سازی و عادی‌سازی.
* توسعه مدل: مدل‌های یادگیری ماشین را با استفاده از ابزارهای داخلی Kubeflow یا ادغام با سایر چارچوب‌های ML مانند TensorFlow، PyTorch و Jupyter توسعه دهید.
* آموزش مدل: مدل‌های قطار را روی داده‌های مقیاس بزرگ با استفاده از قابلیت‌های کوچک‌سازی Kubeflow، از جمله آموزش توزیع‌شده و شتاب GPU، آموزش دهید.
* استقرار مدل: با استفاده از ابزارهای استقرار Kubeflow، از جمله TensorFlow Serving، Seldon، و KFServing، مدل‌ها را جهت تولید مستقر کنید.
* نظارت و نگهداری: مدل‌های در حال تولید، از جمله عملکرد، دقت و جابجایی داده‌ها را پایش کنید و مدل‌ها را در صورت لزوم به‌روزرسانی کنید.
* همکاری: تشویق به همکاری بین دانشمندان داده، توسعه دهندگان و تیم‌های عملیاتی جهت اطمینان از MLOهای روان و مؤثر.
* یکپارچه سازی مداوم و استقرار مستمر (CI/CD): فرآیند یکپارچه سازی، آزمایش و استقرار مدل‌ها را با استفاده از ابزارهایی مانند جنکینز یا GitLab به صورت خودکار انجام دهید.
* بررسی و بهینه‌سازی: به طور مداوم پیاده‌سازی Kubeflow از جمله عملکرد، مقیاس‌پذیری و ادغام با ابزارها و خدمات دیگر را بررسی و بهینه کنید.

# تحلیل هزینه

1. Model drift [↑](#footnote-ref-1)
2. <https://www.samsungsds.com/vn/ai-kubeflow/kubeflow.html> [↑](#footnote-ref-2)
3. Data Lakes [↑](#footnote-ref-3)
4. Data WareHouses [↑](#footnote-ref-4)